ANALYSIS OF RADIATIVE AND CONVECTIVE ENERGY TRANSPORT PROCESSES IN ULTRAHIGH TEMPERATURE VAPOR CORE REACTORS

By

SE-WON CHUNG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1992

To My Parents

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his advisor and supervisory committee chairman, Dr. Samim Anghaie, for his invaluable inspiration, encouragement, and advice. Without his guidance and support, this work could never have been completed. He is also grateful to all the other members of the supervisory committee, Dr. Nils J. Diaz, Dr. Edward T. Dugan, Dr. William E. Lear, and Dr. Robert J. Hanrahan, for their assistance and advice regarding this dissertation.

He would like to thank sincerely all the members and good friends of the Innovative Space Power and Propulsion Institute for their helpful discussions and assistance. Special thanks are due to Mr. Gary Chen who spent numerous hours developing the plotting package used in the documentation of the results. He also acknowledges Mr. Jim White for correcting the manuscript of this dissertation.

Finally, his sincere gratitude goes to his family, especially his wife and children for their sacrifice and understanding in supporting him throughout this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

ANALYSIS OF RADIATIVE AND CONVECTIVE ENERGY TRANSPORT PROCESSES IN ULTRAHIGH TEMPERATURE VAPOR CORE REACTORS

By

Se-Won Chung

August 1992

Chairperson: Samim Anghaie
Major Department: Nuclear Engineering Sciences

Axisymmetric, thin-layer Navier-Stokes equations with radiative heat transfer and internal heat generation terms are used to describe the thermal and flow fields of interest to Ultrahigh Temperature Vapor Core Reactor systems. An implicit-explicit, finite volume method in conjunction with an algebraic two-layer eddy viscosity turbulence model is used to numerically solve these equations. Algebraic grid clustering technique is utilized to resolve the thin boundary layer near the solid wall. Convective heat fluxes into the solid wall are predicted by Fourier's law in the highly stretched grid systems. Radiative heat transport process is modeled using both the 1-D integral approximation and the Rosseland's diffusion approximation. Using a straight tube geometry and the converging-diverging geometry of the reactor exit nozzle, the convective and radiative

heat transfer rates are calculated. Existing experimental data for high temperature air flow in tubes are used to benchmark the model. Good agreement between the numerical results and the measured data indicates the validity of the developed method for the intended analysis.

Results of the analysis indicate that the accurate prediction of convective heat transfer rates at the wall requires grid sizes less than the thickness of the laminar sublayer. For the converging-diverging nozzle, the maximum value of convective and radiative heat fluxes occurs just upstream to the throat of the nozzle where the mass flux is maximum. In general, the convective heat flux increases with increasing the stagnation pressure which corresponds to larger mass fluxes. It is also found that the diffusion approximation for the calculation of radiative heat flux is only valid for an optically thick gas. For less opaque gas conditions the integral transport approximation is applied. The radiative heat transfer rate is dominant at low pressures and high gas temperatures.

CHAPTER 1
INTRODUCTION

1.1 Motivation and Background

Of the numerous space nuclear power concepts proposed and studied in recent years, a UF_4-fuel based vapor core reactor (UTVR) with magnetohydrodynamic (MHD) generator is one of the most promising and interesting. The reactor concept is employed for the analysis of fluid flow and heat transfer problems in this study. The analysis is especially focused on parts of the UTVR system where the maximum heat loading to the wall is expected.

Figure 1.1 shows the basic concept of ultrahigh temperature vapor reactor with advanced energy conversion system. In this system, the average exit temperature range for the core is between 4000-5000 K and in the MHD duct is between 2100-2500 K. The inner wall temperatures of the core and the MHD generator are around 2000 K [1].

Since a very high temperature working fluid and fuel are used to enhance the efficiency of the energy conversion system, potential structural and thermal design problems arise due to the high heat transfer expected to occur between the gas and the solid walls of the core and the nozzle. Accordingly, the accurate prediction of local wall

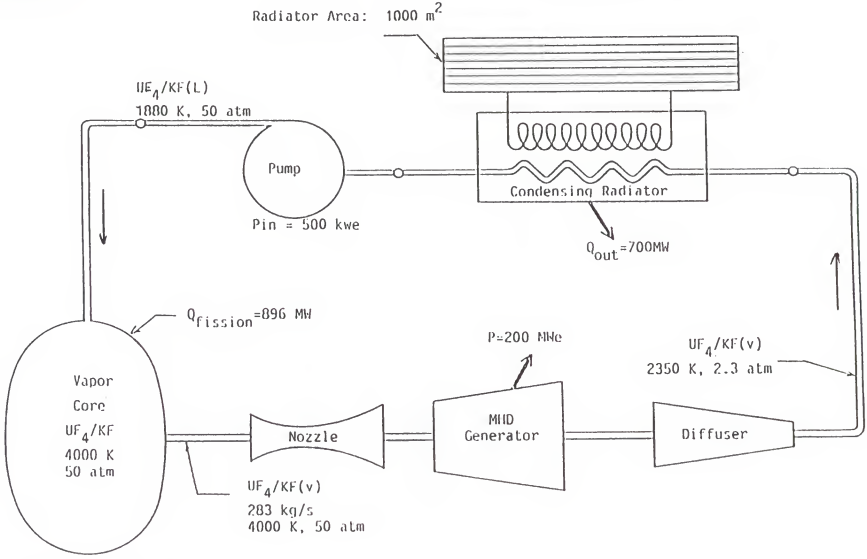


Figure 1.1 200 MWe Ultrahigh Temperature Vapor Reactor-MHD Generator Rankine Space Power System

heat transfer rates is of crucial importance in order to prevent a catastrophic failure, such as a deterioration of wall structural integrity. The energy deposition to the solid wall can be made up of fission fragment heating, β and γ ray heating, radiative heat transfer, and convective wall heat transfer. However, only the convective and radiative heat transfer mechanisms, which dominate the other heat transfer mechanisms, are considered in this study.

Prediction of the convective heat transfer rate in complex flow fields similar to those of the UTVR-MHD system is cumbersome. Moreover, the radiative heat transfer process due to a high temperature flowing medium is another energy transfer mechanism which adds to the complexity of the problem. Consequently, a consistent method of predicting the convective and radiative heat transfer rates must be developed. This gives a strong motivation for the present work.

Prediction of convective heat transfer rates has been a major area of research for several decades in heat transfer. Initially, dimensional analysis was used to develop empirical correlations for heat transfer coefficients. However, these empirical correlations are limited by the system geometry and the fully developed flow condition.

For a fully-developed turbulent flow in a tube, Dittus-Boelter [2], Colburn [3], and Sieder and Tate [4] developed a variety of empirical correlations. These investigations

primarily used the data from experiments performed in long tubes under low temperature flow conditions at the ambient pressure. Later, Zellnik and Churchill [5] extended the temperature range of the flowing medium, and partially considered the entrance effect. However, the limitations of the above methods still exist for geometries and flow conditions.

With the advent of fast computers, computational fluid flow and heat transfer models, which can be applied to every possible flow geometry and condition, have been developed and refined. However, there are still many problems to be solved for the prediction of wall heat transfer by computational heat transfer modeling.

In particular, previous computational fluid flow and heat transfer modeling for the wall heat transfer prediction have focused on the question of providing detailed near-wall resolution. Viegas, Rubesin, and Horstman [6] have addressed the difficulty of dealing with the numerical computation of wall heat transfer prediction. Thus, they have employed the empirical or semiempirical information concerning the nature of the turbulent boundary layer near a surface instead using a fine near-wall grid.

Other researchers used the empirical or semi-empirical information, that is, "wall functions" for their heat transfer studies [7-10]. Rose [11] also used the wall function in conjunction with an algebraic turbulence model

in his heat transfer modeling. However, a recent report [12] pointed out the disadvantages of the wall function method, which are too sensitive to the near wall grid position, and the inaccuracy at the region of flow separation and reattachment.

In this report a transfer model is presented based upon a numerical patching technique in which two different sets of governing equations for a boundary layer region and an outer flow region are set up, respectively. In the patching method, the boundary layer solution is coupled to the outer flow solution through a matching condition applied at the outer edge of the wall layer. This method improves computational efficiency by using a coarse grid in the outer layer region and a very fine grid in the wall layer region.

As a result, wall heat flux and wall shear stress are calculated directly from the slopes of the calculated velocity and temperature profiles at the wall. However, the patching technique is not appropriate for the nozzle flow which is characterized by strong viscous/inviscid, shock/boundary layer interactions.

It is, therefore, required to use an approach which solves a suitable subset of Navier-Stokes equations since this method considers these interactions in a fully coupled manner. To use this approach for prediction of heat transfer rate to the wall, a highly-stretched fine grid near the solid wall is required. The fine mesh, however, yields

computational difficulties in terms of the stability limitation and the computational time. It is logical then to explore numerical methods which would relieve the severities of these limitations.

The numerical solution which requires fine spatial mesh distributions to resolve viscous layer near-wall boundaries is subject to the small time step limitation for the explicit algorithm. To remove the time step limitation, the fully implicit numerical schemes were developed in the mid-1970's by Briley and McDonald [13] and Beam-Warming [14]. Although the implicit approximate factorization procedure can have a larger time step than the explicit procedure, the factorization introduces an error proportional to $(\Delta t)^2$ which restricts the time step from being too large.

An hybrid implicit-explicit method presented by McCormack [15,16] requires the simple inversion of block bidiagonal matrices and, hence, greatly reduces the computing time needed. This numerical scheme with Baldwin and Lomax's algebraic turbulence model [17] is utilized in this study.

In addition, radiative heat transfer caused by the high temperature fuel-working fluid becomes another important transport mechanism in this system. In the feasibility study of the direct flow gaseous core reactor system [18], a diffusion approximation for an optically dense gas appeared to be a good approximation as well as an integration method

of the radiation equation. Other researchers [19-22] also used the diffusion approximation and mentioned that the method was valid in the optically thick medium, that is, the mean penetration distance of photon is quite small compared to the characteristic dimension of the medium.

With the above considerations, the model, which solves a suitable subset of Navier-Stokes equations with radiative heat transfer and an internal heat source term in a very fine near-wall grid, is regarded as one of the best models for this system. This information, based on the previous studies, gives the direction of modeling to fit the present flow phenomena and problem characteristics. In the following section, the objectives of the present work and a preview of the mathematical model will be described in more detail.

1.2 Objectives and Preview of Mathematical Model

The primary objective of this work is to develop a computational fluid dynamics and heat transfer model for flow of a very high temperature radiating gas with internal heat generation in a UTVR/MHD system. In particular, to provide a methodology for accurate estimation of heat transfer rates from the hot gas to the inner wall of the reactor chamber and associated nozzle is a primary concern of this study.

To achieve this goal, a two-dimensional (2-D), axisymmetric Navier-Stokes equations solver along with the

turbulence model is used. A near-wall diffusion model for convective wall heat transfer and Rosseland's diffusion model and 1-D integral approximation for radiative heat transfer are also employed.

The procedure is based on the solution of Navier-Stokes equations for unsteady, compressible, viscous, turbulent flow. To obtain the flow and thermal field solution, the axisymmetric thin layer Navier-Stokes equations with radiative heat transfer and internal heat source are used and numerically solved.

A hybrid implicit-explicit method in conjunction with Lomax and Baldwin's two-layer algebraic turbulence model is used, and the finite volume approach is also applied to this analysis. A flux-splitting procedure [14], which takes advantage of the natural direction of information travel and enables the implicit coefficient matrix to be more diagonally dominant, is employed. The Rosseland's diffusion approximation is employed to account for the radiation contribution, which models the radiation heat transfer in the partial differential equation, instead of the partial integro-differential equation. The 1-D integral approximation is also used for the radiative heat transfer.

For prediction of convective heat transfer rate, a highly stretched grid system, in which several grid points can be located in the viscous sublayer, is used. Thus, the convective heat transfer rate to the wall is directly

determined by the known temperature gradient at the wall. With the developed model, the effect of internal heat generation on the fluid flow and heat transfer in a straight tube is investigated. The effect of numerical grid characteristics, i.e., skewness and orthogonality, is also examined in the flow and thermal field solutions. In addition, more computations are carried out for a variety of flow conditions, geometric parameters, radiative mean opacity, and constant power densities.

1.3 Overview

Theoretical modeling, including the thin layer Navier-Stokes equation, diffusion model for radiative heat transfer and turbulence model, is described in Chapter 2. Chapter 3 provides an algebraic stretching grid scheme for a converging-diverging nozzle and also explains the algebraic method for obtaining an orthogonal grid system in the nozzle. Then in Chapter 4 a numerical algorithm and boundary conditions are described for the solution of the governing equations. Chapter 5 provides a method for evaluating the heat transfer using the known temperature profile. Finally, Chapter 6 presents results and Chapter 7 provides conclusions.

CHAPTER 2
MATHEMATICAL FORMULATION

In this chapter the governing equations for unsteady compressible flow with thermal radiation and internal heat generation are summarized. Based on the physical considerations as well as computer limitations, the thin-layer Navier-Stokes equations are used for this study. The two-dimensional (2-D) axisymmetric thin-layer Navier-Stokes equations are expressed in conservation law form. The diffusion model for radiant heat transfer and the Baldwin-Lomax turbulence model are described.

2.1 Governing Equation

In modeling the simultaneous development of momentum and thermal boundary layer of a radiating gas in the axisymmetric cylindrical coordinate system (z,r), the following assumptions are made:

(1) Viscous dissipation, heat conduction, and thermal radiation in the streamwise direction are negligible.

(2) Optical properties of the participating gas are approximated in terms of those of a grey emitting, absorbing, and nonscattering gas.

(3) There is no line radiation at temperatures considered for this analysis.

(4) When the gas properties at high temperature are not available, the perfect gas law is applied to evaluate them.

(5) There is no dissociation or phase change for UF_4 gas in the 1800 to 4000 K temperature range. This assumption is based on the following experimental data taken from Reference [23].

(a) Dissociation of UF_4 is less than 1% at 1 atm and 3500 K.

(b) Dissociation of UF_4 is less than 1% at 100 atm and 5000 K.

Under these assumptions, the differential equations used to describe the mean flow for this study are the time-dependent, mass-averaged Navier-Stokes equations with radiative and constant internal heat source terms for the axially symmetric flow of a compressible fluid. Utilizing the Boussinesq assumption for the turbulence model, the resulting equations [24, 25], in strong conservation form, can be written as follows:

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}_1}{\partial z} + \frac{\partial \vec{G}_1}{\partial r} + \vec{H}_1 = \frac{\partial \vec{G}_v}{\partial r} + \vec{H}_v \qquad (2.1)$$

in which

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \vec{F}_i = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e + p)\, u \end{bmatrix}, \quad \vec{G}_i = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (e + p)\, v \end{bmatrix}, \quad \vec{H}_i = \frac{1}{r}\begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ (e+p)\, v \end{bmatrix} \tag{2.2}$$

and the viscous terms $\vec{G}_v$ and $\vec{H}_v$ are

$$\vec{G}_v = \begin{bmatrix} 0 \\ \mu_T \dfrac{\partial u}{\partial r} \\ 4\dfrac{\mu_T}{3}\dfrac{\partial v}{\partial r} - \dfrac{2}{3}\mu_T \dfrac{v}{r} \\ \mu_T u \dfrac{\partial u}{\partial r} + 4\dfrac{\mu_T}{3} v \dfrac{\partial v}{\partial r} - (q_c'' + q_r'') \end{bmatrix} \tag{2.3}$$

and

$$\vec{H}_v = \frac{1}{r}\begin{bmatrix} 0 \\ \mu_T \dfrac{\partial u}{\partial r} \\ 2\mu_T\left(\dfrac{\partial v}{\partial r} - \dfrac{v}{r}\right) \\ \mu_T u \dfrac{\partial u}{\partial r} + \dfrac{4}{3}\mu_T v \dfrac{\partial v}{\partial r} - (q_c'' + q_r'') + \dot{Q} \end{bmatrix}. \tag{2.4}$$

Here, ρ is the density; u and v are the velocity components in the z and r directions, respectively; $\dot{Q}$ is the uniform internal heat generation term; q''_c and q''_r are conductive

and radiative heat fluxes, respectively; p is the pressure; and e is the total energy per unit volume which is related to the internal energy ε and the kinetic energy. The total energy per unit volume is written by

$$e = \rho \left[\varepsilon + \frac{1}{2} (u^2 + v^2) \right]. \tag{2.5}$$

The total viscosity μ_T is given as the sum of its molecular, μ_l, and eddy components, μ_t:

$$\mu_T = \mu_l + \mu_t . \tag{2.6}$$

The molecular viscosity for air is assumed to obey Sutherland's formula [25]:

$$\mu_l = C_1 \frac{T^{3/2}}{(T + C_2)} \tag{2.7}$$

where C_1 and C_2 are constants for a given gas. For air at moderate temperature, $C_1 = 1.458x10^{-6}$ $(kg/m\ s\ \sqrt{K})$ and $C_2 = 110.4\ K$. The turbulent eddy viscosity is obtained from the turbulent eddy viscosity model proposed by Baldwin and Lomax. The heat flux terms are comprised of the conductive heat flux term, q_c'', and the radiative heat flux term, q_r''. The conductive heat flux can be written as

$$q_c'' = -k_c \frac{dT}{dy} \tag{2.8}$$

where

$$k_c = c_p \left(\frac{\mu_l}{Pr_l} + \frac{\mu_t}{Pr_t} \right) \tag{2.9}$$

Typical values of the molecular and turbulent Prandtl numbers for air at standard conditions are 0.9 and 0.72, respectively. The radiative heat flux term is presented in the following section.

The complete set of time-dependent thin-layer Navier-Stokes equations (2.1) includes one continuity equation, two momentum equations, and one energy equation, but there are six unknowns, ρ, u, v, e, p, and T. In order to close the system of equations, the fluid is assumed to be a perfect gas. The equation of state and other relations for a perfect gas are

$$p = \rho RT, \quad \varepsilon = c_v T, \quad h = c_p T, \quad \gamma = \frac{c_p}{c_v} \tag{2.10}$$

where R is the gas constant, h is enthalpy, and c_p and c_v are the specific heat at constant pressure and volume, respectively. γ is the ratio of specific heats, which for air at standard conditions is 1.4. Using the above formula, (2.5) and (2.10), two additional equations relating p and T

are obtained as follows:

$$p = (\gamma-1)\left[e - \frac{1}{2}\rho\,(u^2 + v^2)\right]$$
$$T = \frac{(\gamma - 1)}{R}\left[\frac{e}{\rho} - \frac{1}{2}\,(u^2 + v^2)\right]. \qquad (2.11)$$

Hence, equations (2.1) and (2.11) are the governing equations for axisymmetric, unsteady compressible flow problems. In this study, two sets of fluid properties are used. One is air at standard conditions for the verification purpose of fluid flow and heat transfer solutions, and the other one is UF_4 gas properties evaluated at 2000 K. The properties of the UF_4 gas are presented in the following subsection.

2.1.1 UF_4 Properties

As mentioned above, two sets of fluid properties are utilized to analyze the fluid flow and heat transfer problems in this work. One is the air and the other is pure UF_4 gas which is described in this section. Pure UF_4 gas is treated as a perfect gas and is assumed not to be dissociated at temperatures between 1800 K and 4000 K. The properties of pure UF_4 gas are summarized in Table 2.1 and Table 2.2. The specific heats and the ratio of specific heats (γ) data are taken from Reference [26]. The viscosity

Table 2.1 Pure UF_4 properties at 2000 K.

M_w	(Kg/Kg·mol)	311.15
R	(J/Kg·K)	26.72
γ		1.08
C_p	(J/Kg·K)	345.95
C_v	(J/Kg·K)	319.31
μ_l	(Kg/m·s)	8.67×10^{-5}
K_l	(W/m·K)	0.028
Pr		0.915

Table 2.2 Heat capacities of UF_4 gas at temperature between 1800 and 4000 K.

Temperature in K	C_p in J $Kg^{-1} \cdot K^{-1}$	C_v in J $Kg^{-1} \cdot K^{-1}$	Ratio of Specific Heat, γ
1800	345.68	318.90	1.08
2000	345.95	319.31	1.08
2200	346.22	319.58	1.08
2400	346.29	319.71	1.08
2600	346.62	319.98	1.08
2800	346.76	320.12	1.08
3000	346.89	320.16	1.08
3200	347.03	320.25	1.08
3400	347.03	320.25	1.08
3600	347.16	320.38	1.08
3800	347.16	320.38	1.08
4000	347.16	320.52	1.08

of pure UF_4 gas is assumed to be the same as that of UF_6 reported by Oliver and Dugan [27]. All properties are estimated at 2000 K. Indeed, it matters little in this work what particular gas properties are employed, because the primary concern of this study is not in the application of a design problem but in providing a robust and accurate heat transfer model. Using the known ratio of specific heats, γ, a laminar Prandtl number is determined by Euken's formula [28]

$$Pr = \frac{4\gamma}{9\gamma - 5} \quad . \tag{2.12}$$

The Prandtl number is often used to determine the coefficient of thermal conductivity k_e once the viscosity is known.

2.2 Radiative Heat Transfer

The thin-layer Navier-Stokes equations in conjunction with the Baldwin-Lomax algebraic turbulence model were expressed and used to solve flow equations in a coupled manner with the energy equation. Internal heat generation and radiative heat transfer are considered in the energy equation as source terms, $\dot{Q}$ and $\nabla \cdot q_r''$, respectively. In the present study, the internal heat source term is assumed to be constant; however, modeling or some simplification for radiative heat transfer is needed. The assumption of opaque

gas can lead to a substantial simplification such as a diffusion approximation. In this diffusion approximation, the radiation arriving at any location comes only from the immediate surroundings because any other radiation is absorbed before arriving at that location. The diffusion approximation with the above assumption leads to a considerable simplification in the expression for radiative heat flux, but the simplicity is offset by its disadvantage. For instance, the approximation is valid at points far from the solid surface and only for intense absorption, i.e., when the fractional variation in temperature is small in a distance of one mean free path. Additionally, the approximation breaks down in the vicinity of the surface, since it does not take into account radiation leaving from the surface. Therefore, the limit and the accuracy of the approximation should be examined. For comparison purposes, the radiative heat flux is predicted by two different approximations, the diffusion approximation and the one-dimensional (1-D) equation of radiative transfer. The two approximations are described below.

2.2.1 Diffusion Approximation

As mentioned above, the diffusion approximation is only valid for an optically dense medium. This can be stated more rigorously by letting H be a path length over which the radiant energy density does change appreciably, and letting

l_m be the extinction mean free path. Then, the diffusion approximation is applicable for the condition of $l_m/H << 1$. Under the assumption of an optically thick medium, the radiative heat flux is proportional to the temperature gradient and is written as [29]

$$q_r'' = -\frac{4}{3a_R}\nabla e_b = -\frac{16\sigma T^3}{3a_R}\nabla T = -K_r\nabla T \tag{2.13}$$

where K_r, the radiative conductivity, is defined by

$$K_r = \frac{16\sigma T_w^3}{3a_R} \tag{2.14}$$

where a_R is the Rosseland mean opacity and σ is the Stefan-Boltzmann constant. To obtain the mean opacity, the spectral absorption coefficient must be averaged over the spectrum. Consequently, the Rosseland mean opacity of a nonscattering gas is defined by [29]

$$a_R = \frac{\int_0^\infty \frac{\partial B_\nu}{\partial T}d\nu}{\int_0^\infty \frac{n^2}{a_\nu}\frac{\partial B_\nu}{\partial T}d\nu} \tag{2.15}$$

where ν is the photon wave number, B_ν is the Planck function, a_ν is the spectral absorption coefficient, and n

is the real index of refraction. Due to the lack of experimental data for the spectral absorption coefficient of UF_4, direct calculation of the Rosseland mean opacity for UF_4 is not feasible. However, for this study where the self-absorption by UF_4 is under consideration, the Rosseland mean opacity can be estimated by [30]

$$a_R = N\sigma_{ph} \tag{2.16}$$

where N is the molecular number density of the gas and σ_{ph} is the photon collision cross section per molecule. Using the assumption of the perfect gas law, the molecular number density can be computed by

$$N = \frac{p}{\kappa T} \tag{2.17}$$

where κ is Boltzmann's constant, p is the gas pressure, and T is the gas temperature. Therefore, the radiative conductivity can be written as

$$k_r = \frac{16\sigma_{SB}\kappa}{3\sigma_{ph}p} T^4 \tag{2.18}$$

The photon absorption cross section for UF_4 gas is very close to that of UF_6 and approximated based on the data

presented in Reference 31. The frequency averaged photon absorption cross section in the 2000 Å to 4200 Å wavelength range for UF_4 is estimated to be equal to the frequency averaged photon absorption cross section of UF_6 which is 2.76 million barns (2.76×10^{-22} m^2). The frequency averaged Rosseland opacity for UF_4 gas based on the estimated photon absorption cross section in the temperature range of 2000 K to 5000 K is

$$a_R = 2x10^4 \frac{P}{T} \qquad (2.19)$$

where p is in atm, T is in K, and a_R is in cm^{-1}. The estimated values of the frequency averaged Rosseland opacity agree with the published values of the mean opacity for pure uranium gas at 5000 K [32]. Using the estimated opacities, the Rosseland approximation leads to a considerable simplification in the expression for radiative heat flux. However, when the frequency averaged Rosseland coefficient is applied to real gases which are usually transparent in some wavelength regions, some errors might be introduced. To assess the sensitivity of radiative transfer to the average value of the Rosseland absorption coefficient, a wavelength-band application is performed for two wavelength-bands, 2000 Å to 3000 Å and 3000 Å to 4200 Å. In addition, the diffusion approximation is not accurate near the solid

boundary since it does not take into account radiation leaving from the surface; the presence of the surface changes the coefficient from 16/3 to 8/3 [33].

2.2.2 Approximation by Using 1-D Equation of Radiative Transfer

This approximation is performed with the integral form of the one-dimensional equation of radiative transfer by using the frequency averaged opacity utilized in the diffusion approximation. With no incident radiation at the center line, the equation of radiative transfer is written as [29]

$$i(r) = \int_0^r a\, I(r') \exp(-a(r-r'))\, dr' \qquad (2.20)$$

where i is the intensity, and a is the frequency averaged Rosseland opacity, and I is the source function, which can be reduced to the local blackbody intensity in absorbing medium without scattering. The local blackbody intensity is given by

$$I = \frac{\sigma T^4}{\pi} \qquad (2.21)$$

In order to simplify the equation (2.20), moreover, it is assumed that the wall is a blackbody, and the radiation crossing a unit area normal to the r direction is only

considered in intensity terms of the positive and negative sense. The net radiative flux in the gas in the positive r direction is

$$q_r'' = q^+ - q^- = \pi(i^+(r) - i^-(r)) \qquad (2.22)$$

where q^+ is the radiative flux traveling from gas to the boundary and q^- is the radiative flux in the opposite direction.

The assumption of considering only the radiation normal to the z direction might possibly be acceptable under certain conditions, that is, the temperature variation in the axial direction is very small and the geometry is very simple, such as a straight tube.

The simplified integral equation shown above, is solved by using the simple trapezoidal rule. At every node point the net radiative flux is calculated by summing the positive values evaluated from the interlayer of the gas to the wall and the negative values for the opposite direction. In this summation, the radiation is considered to be absorbed within a five-times distance of the mean free path.

2.3 Algebraic Turbulence Model

The algebraic turbulence model employed in this study is the one proposed by Baldwin and Lomax [17]. It is a two-layer algebraic eddy viscosity model in which μ_t is

calculated for an inner and an outer region. The inner region follows the Prandtl-Van Driest formulation. In both the inner and outer formulation, the distribution of vorticity is used to determine the length scales, thereby avoiding the necessity of finding the outer edge of the boundary layer. For the inner region,

$$(\mu_t)_{inner} = \rho l^2 |\omega| \tag{2.23}$$

where

$$l = \kappa y[1 - \exp(-y^+/A^+)] \tag{2.24}$$

and

$$y^+ = \frac{\rho_w u_\tau y}{\mu_w} = \frac{\sqrt{\rho_w \tau_w}\, y}{\mu_w} \tag{2.25}$$

Here, y is the normal distance to the surface and $|\omega|$ is the absolute magnitude of vorticity. The nondimensional parameter, y^+, is the law of the wall coordinate. l is the mixing length and u_τ, the frictional velocity, is

$$u_\tau = \sqrt{\frac{\tau_w}{\rho_w}} \tag{2.26}$$

where τ_w, ρ_w and μ_w are the local shear stress, density, and laminar viscosity evaluated at the wall.

The eddy viscosity for the outer region is given by

$$(\mu_t)_{outer} = KC_{cp}\rho F_{wake} F_{kleb}(y) \tag{2.27}$$

where K is the Clauser constant, C_{cp} is an additional constant, and F_{wake} is the smaller value of the following:

$$\begin{aligned} F_{wake} &= y_{max} F_{max} \\ F_{wake} &= C_{wk} y_{max} U_{dif}^2 / F_{max} \end{aligned} \tag{2.28}$$

The quantities y_{max} and F_{max} are determined from the function

$$F(y) = y|\omega|[1 - \exp(-y^+/A^+)] \tag{2.29}$$

In a wake, the exponential term of the above equation is set equal to zero. The quantity F_{max} is the maximum value of F(y) that occurs in a profile, and y_{max} is the value of y at which it occurs. The function $F_{kleb}(y)$ is the Klebanoff intermittency factor given by

$$F_{kleb}(y) = [1 + 5.5(\frac{C_{kleb}y}{y_{max}})^6]^{-1} \tag{2.30}$$

The quantity U_{dif} is given as the difference between maximum and minimum total velocity in the profile

$$U_{dif} = (\sqrt{u^2+v^2})_{max} - (\sqrt{u^2+v^2})_{min} . \qquad (2.31)$$

The minimum total velocity is taken to be zero, unless a wake is considered. The constants appearing in the foregoing relations have been determined by requiring agreement with the Cebeci model for constant pressure boundary layers at transonic speeds. The constants used for this model are $A^+ = 26$, $C_{cp} = 1.6$, $C_{wk} = 0.25$, $C_{kleb} = 0.3$, $\kappa = 0.4$, and $K = 0.0168$.

CHAPTER 3
GRID GENERATION

3.1 Grid for Numerical Solution

To solve the governing equations in differential forms for fluid mechanics and heat transfer, an approximation to the partial differential equations is applied and a numerical solution scheme is introduced. The approximation converts the partial derivatives to finite difference expressions, which are used to rewrite the partial differential equations as algebraic equations. Then the approximate algebraic equations, referred to as finite difference equations, are subsequently solved at discrete points within the domain of interest. Therefore, a set of grid points within the domain, as well as the boundaries of the domain, must be specified. The creation of such a grid system is known as grid generation.

The generation of a grid is one of the central problems in computing numerical solutions. A well constructed grid greatly simplifies the numerical solution of systems of partial differential equations. On the other hand, an improper grid choice may lead to instabilities and lack of convergence.

In general, grid generation techniques may be classified as

(1) algebraic methods,

(2) partial differential methods,

(3) complex variable methods (conformal mapping).

The selection of the most suitable and efficient technique of grid generation for a particular problem depends on the geometry of the physical domain.

In the axisymmetric code used for this study, the algebraic grid generation technique has been adopted. In general, algebraic methods use some special formula for the smooth grid network of a certain geometry. The converging-diverging nozzle, which is a critical component in the UTVR-MHD power system concept, with large momentum and energy transfer gradients is considered in this study. The algebraic grid generation technique for this geometry is briefly explained in the following.

The general shape of a converging-diverging nozzle is shown in Figure 3.1. First, the coordinate of every grid point between the center line and the solid wall at the throat is calculated by the following formula:

$$r(j) = -R\left(1 - \frac{\exp[C_k \delta (j-2)] - 1}{E_k}\right) \quad , \quad j = 2, \ldots\ldots, Jmax \qquad (3.1)$$

where the parameters, C_k and E_k, which are determined in iterative procedures, control the degree of the density or the sparsity of the grid system. Another parameter, δ, is

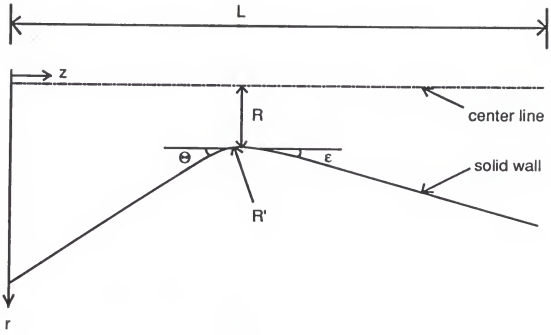


Figure 3.1 Geometry of a nozzle with a curvature at throat

the uniform grid interval at the throat for which the throat radius R is divided by the number of radial direction grids. Then, the coordinates of grid points on the solid wall are algebraically determined by the uphill angle, Θ, the downhill angle, ε, and the curvature, R', at the throat. The coordinate of each grid point in the radial direction is determined by the distance ratio of every grid interval between the consecutive grids to the throat radius at the throat. In the same manner, all the interior grid coordinates along the flow stream direction are calculated.

on the other hand, the physical coordinate grid must be finer near the walls and through the regions of major disturbances, such as the expansion region. Well away from these regions a coarser physical grid can be used for computational efficiency, while still adequately resolving the flow field. The grid points along the radial direction are clustered toward the rigid flow boundaries, where large gradients are expected, using exponential stretching. The degree of stretching along the radial direction are controlled by the parameters, C_k and E_k, used in Eq. (3.1). The parameters are determined by the analytical formulas

$$F_k = dy - R\frac{\exp(C_k\delta) - 1}{\exp(C_k) - 1} \qquad (3.2)$$

and

$$F_{kp}=-R\frac{(\exp(C_k)-1)\,\delta\exp(c_k)-(\exp(C_k\delta)-1)\exp(C_k)}{(\exp(C_k)-1)^2} \qquad (3.3)$$

C_k is the last updated value for the specified number of iteration as follows:

$$C_k=C_k-\frac{F_k}{F_{kp}} \qquad (3.4)$$

E_k is evaluated using the determined C_k in the following:

$$E_k=\exp(C_k)-1.0 \qquad (3.5)$$

Attention must be paid to the manner in which the grid points are distributed along the axis because the distribution also affects to a large extent the accuracy of the calculations. The grid points along the axis must be concentrated in the areas of large wall slopes and axial locations where separation is expected to occur. The same grid-clustering scheme is applied to the axial direction as used in the radial direction. The physical domain grid constructed using Eqs. (3.1) , (3.2), (3.3), (3.4), and (3.5) is shown in Figure 3.2.

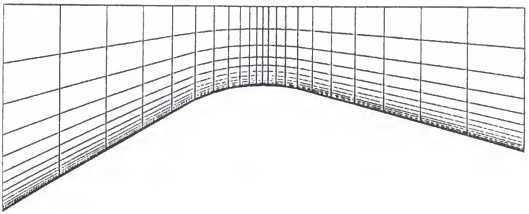


Figure 3.2 A skewed grid in a converging-diverging nozzle.

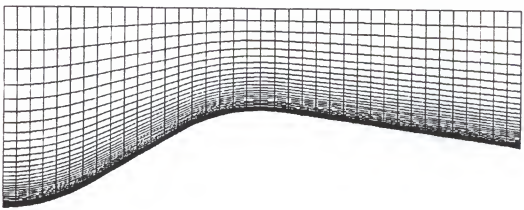


Figure 3.3 A skewed grid in a bell-type nozzle.

The above basic algebraic technique can also be applied to a different type of nozzle, such as a bell-type nozzle, with minor modification. The uniform grid system for the bell-type nozzle is presented in Figure 3.3. Both types of nozzles are used in this work for the study of flow and heat transfer in the nozzle.

3.2 Orthogonal Grid with an Algebraic Method

The algebraic method for grid generation is getting popular due to its simplicity and the computational speed it provides. However, the algebraic grid system usually has significant skewness which is the main disadvantage of the algebraic method. Severe skewness must be avoided because it creates problems with the numerical implementation when a generalized coordinates transformation is used to transform the physical domain grid to the computational domain grid.

In the following a simple algebraic method is presented to transform a skewed grid to an orthogonal grid. The method is applied to the converging-diverging nozzle. The algebraic grid generation technique explained in the previous section does a piecewise linear approximation of the boundary curve, S_1S_2, as shown in Figure 3.4. From each axial position along the boundary curve, a vertical line to the symmetry axis is drawn. Along each vertical line the grid points are distributed closer to the boundary curve using the exponential clustering scheme.

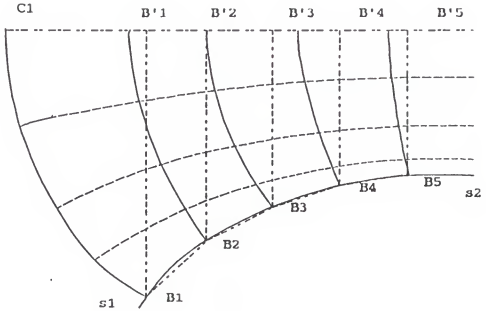


Figure 3.4 Skewed grid(dotted line) and corrected grid(solid line)

Figure 3.4 shows that this approach results in severely skewed grids when the slope of the boundary curve with respect to the symmetry axis is large. In order to correct the grid skewness, it is necessary to draw a curve which is normal on both the boundary curve and the axis of symmetry.

For the particular case where the symmetry axis is a straight line the curve is a section of a circle. The center lies at the intersection of the axis with the straight line segment, B_1B_2, approximating the curved boundary. The radius of the circle is OB_1, which is the distance between the interaction point O of B_1B_2 and the symmetry axis. This circular sector, indeed, intersects at right angles with both the boundary curve and the axis.

The grid points on the vertical line presented in Figure 3.5 are now shifted to new positions on the circular arc. The new positions are the intersections of the circular arc and the family of lines emanating from the center O and passing through the old positions on the vertical. The radius of the circle and the new shifted coordinates of the grid points are calculated from the following trigonometric relations:

$$\begin{aligned} \alpha &= \tan^{-1}\left[\left(B_1B_1' - B_2B_2'\right) / B_1'B_2'\right] \\ OB_1 &= B_1B_1' \frac{R}{\sin\alpha} \\ \beta &= \tan^{-1}\left(P_1B_1' / OB_1'\right). \end{aligned} \tag{3.6}$$

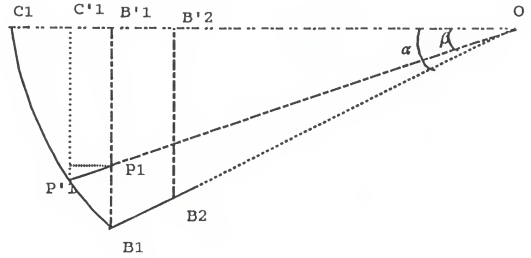


Figure 3.5 Displacement of grid points from the old position P1 to the new position P'1 (corrected grid)

The modified orthogonal grids for the above two types of nozzle are shown in Figures 3.6 and 3.7, respectively. The improvement in terms of orthogonality is observed from a visual comparison of the skewed and the orthogonal grid or from a comparison of the numerical values of the interaction angles of the skewed and normalized grids given in Table 3.1.

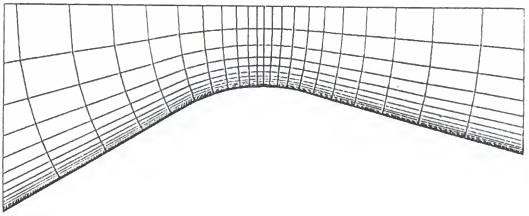


Figure 3.6 Corrected grid in a converging-diverging nozzle.

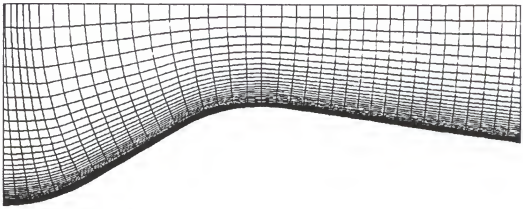


Figure 3.7 Corrected grid in a bell-type nozzle.

Table 3.1 Intersection angles of the skewed grid and the normalized grid in the nozzle of Figures 3.2 and 3.6, along a mesh line close the curved rigid boundary where the maximum skewness is observed.

Skewed Grid	Orthogonal Grid
60°	90°
60°	90°
60°	90°
60°	90°
65°	90°
73°	90°
78°	90°
83°	90°
85°	90°
88°	90°
90°	90°
89°	90°
88°	90°
85°	90°
83°	90°
80°	90°
77°	90°
76°	90°
75°	90°
75°	90°
75°	90°
75°	90°

CHAPTER 4
NUMERICAL ALGORITHMS

In this chapter the finite volume discretization procedure which has been followed to obtain a suitable algorithm for computer implementation, is presented. A hybrid implicit-explicit numerical method based on a flux splitting scheme is described. Finally, boundary and initial conditions and the convergence criterion are discussed.

4.1 Finite Volume Discretization

Finite volume methods approximate integral forms of conservation equations on finite cells. The equations are approximated by summing the fluxes of mass, momentum, and energy from neighboring cells into each cell, thereby satisfying the conservation equations over that cell. In this method, the integral formulation of the conservation laws are discretized directly in the physical domain which allows the use of the shock-capturing technique in the transonic and supersonic flow regions.

Another feature of the finite volume approach is the decoupling of the discretization of the equations from the grid generation so that the rigorous, smooth variation of

mesh size and an explicit coordinate transformation are not required [34]. In order to help the understanding of the process, consider an element located at (i,j) with its four boundaries ab, bc, cd, and da, as shown in Figure 4.1. The integral form of the governing equation given in equation (2.1) can be written as

$$\int_v \left(\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}_1}{\partial z} + \frac{\partial \vec{G}_1}{\partial r} + \vec{H}_1\right)_{i,j} dV_{i,j} = \int_v \left(\frac{\partial \vec{G}_v}{\partial r} + \vec{H}_v\right)_{i,j} dV_{i,j} \tag{4.1}$$

where

$$dV_{i,j} = (r_{i,j}\Delta\theta)\, dA_{i,j}$$

and where Θ is the azimuthal angle and A represents the area which is positive for counter-clockwise elements and negative for clockwise elements. Here, consider the left-hand side, i.e., the inviscid terms, such that the equation is reduced to the Euler equation. Application of Gauss's theorem yields

$$\frac{\partial}{\partial t}\int_{A_0} U\, dA_{i,j} + \int_{A_0} \left(\frac{\partial F}{\partial z} + \frac{\partial G}{\partial r}\right) dA_{i,j} + H_{i,j}A_{i,j} = 0 \tag{4.2}$$

To express the above integral form to a discrete form, let

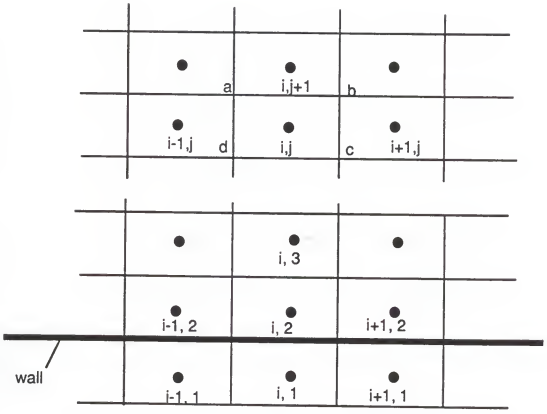


Figure 4.1 A finite volume mesh.

us consider a simple control volume described in Figure 4.1 and a cell (i,j). Then, equation (4.2) can be reduced to the discrete form:

$$\Delta(\vec{U})^n_{i,j} = -\frac{\Delta t}{A_{i,j}}(D_x \vec{E}^n \Delta r + D_x \vec{F}^n \Delta z) + H^n_{i,j} \Delta t \qquad (4.3)$$

where $D_x(\vec{E})_{i,j}$ and $D_x(\vec{F})_{i,j}$ are the increments of inviscid flux terms which can be approximated by forward difference

$$\begin{aligned} D_+(\vec{E})_{i,j} &= (\vec{E})_{i+1,j} - (\vec{E})_{i,j} \\ D_+(\vec{F})_{i,j} &= (\vec{F})_{i,j+1} - (\vec{F})_{i,j} \end{aligned} \qquad (4.4)$$

and backward difference

$$\begin{aligned} D_-(\vec{E})_{i,j} &= (\vec{E})_{i,j} - (\vec{E})_{i-1,j} \\ D_-(\vec{F})_{i,j} &= (\vec{F})_{i,j} - (\vec{F})_{i,j-1} . \end{aligned} \qquad (4.5)$$

The superscript n to indicate present time level is dropped for convenience.

4.2 Implicit-Explicit Scheme

The hybrid implicit-explicit method is employed to solve the governing equation given in equation (2.1). The

basic algorithm of the method is based on the explicit predictor-corrector scheme proposed by MacCormack [35]. The explicit formulation is

$$\Delta U^n_{i,j} = -\Delta t \left[\frac{D_\pm F_i}{\Delta z} + \frac{D_\pm (G_i - G_v)}{\Delta r} - H \right]^n_{i,j} \qquad (4.6)$$

where $\vec{H} = \vec{H}_v - \vec{H}_i$ and $D_\pm$ represents either forward or backward and the arrows indicating vector quantities are omitted. The above explicit algorithm is numerically stable for Courant-Friedrich-Lewy (CFL) numbers less than or equal to one. To remove the CFL restriction, an implicit procedure is required.

Modern implicit numerical methods are of the form [16]

$$[NUMERICS]\ \delta U^{n+1}_{i,j} = [PHYSICS].$$

The above formula calculates the change in the solution $\delta U^{n+1}_{i,j}$ at grid point i,j during the interval from time $n\Delta t$ to time $(n+1)\Delta t$. The term on the right-hand side (physics) represents a finite difference or volume approximation to the complete governing equation (2.1) using the solution at time $n\Delta t$, i.e.,

$$[PHYSICS] = \Delta U^n_{i,j}$$

$$= -\Delta t \left[\frac{D_x F_i}{\Delta z} + \frac{D_x (G_i - G_v)}{\Delta r} - H \right]^n_{i,j} .$$

To derive an implicit form, equation (2.1) is differentiated with respect to time as

$$\frac{\partial (\frac{\partial U}{\partial t})}{\partial t} = -\frac{\partial A_i (\frac{\partial U}{\partial t})}{\partial z} - \frac{\partial (B_i - B_v)(\frac{\partial U}{\partial t})}{\partial r} + \frac{\partial H}{\partial t} \qquad (4.7)$$

where $A_i = \frac{\partial F_i}{\partial U}$, $B_i = \frac{\partial G_i}{\partial U}$ and $B_v = \frac{\partial G_v}{\partial U}$ are the Jacobians of F_i, G_i and G_v with respect to U.

Letting

$$\Delta t \, (\frac{\partial U}{\partial t})^n = \Delta U^n, \qquad \Delta t \, (\frac{\partial H}{\partial t})^n = \Delta H^n$$

and

$$\Delta t (\frac{\partial U}{\partial t})^{n+1} = \delta U^{n+1},$$

an implicit difference approximation to equation (4.7) is

$$\left[I + \Delta t \frac{D \cdot A_i}{\Delta z} + \Delta t \frac{D \cdot (B_i - B_v)}{\Delta r}\right]_{i,j} \delta U^{n+1}_{i,j}$$

$$= \Delta U^{n}_{i,j} + \Delta H^{n}_{i,j}$$

where the dots indicate that the difference operator D applies to all factors to the right of the operator, i.e., to the factor $\delta U^{n+1}_{i,j}$.

To improve numerical efficiency and to take advantage of natural directions in which information propagates, the Steger and Warming's flux vector splitting technique [36] is used. Since the inviscid flux terms which represent the macromotion of working fluid are homogeneous of degree one in the element of U, Steger and Warming pointed out that

$$F_i = A_i U$$
$$G_i = B_i U$$

The Jacobian matrices A_i, B_i can be diagonalized by similarity transformations S_x, S_y such that

$$A_1 = S_x^{-1}\Lambda_A S_x$$

$$B_1 = S_y^{-1}\Lambda_B S_y$$

where

$$S_x = \begin{bmatrix} 1 & 0 & 0 & -1/c^2 \\ 0 & \rho c & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -\rho c & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ \alpha\beta & -u\beta & -v\beta & \beta \end{bmatrix}$$

$$S_y = \begin{bmatrix} 1 & 0 & 0 & -1/c^2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \rho c & 1 \\ 0 & 0 & -\rho c & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ \alpha\beta & -u\beta & -v\beta & \beta \end{bmatrix}$$

and

$$\Lambda_A = \begin{bmatrix} u & 0 & 0 & 0 \\ 0 & u+c & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u-c \end{bmatrix}, \quad \Lambda_B = \begin{bmatrix} v & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v+c & 0 \\ 0 & 0 & 0 & v-c \end{bmatrix}.$$

Here, c is the speed of sound, $\alpha = (u^2 + v^2)/2$, $\beta = \gamma - 1$, and γ is the ratio of the specific heats of gas. The Λ_A and Λ_B are diagonal matrices, and the diagonal elements in these matrices are the eigenvalues of the original Jacobian Matrices A_1, B_1. In general, some of the elements of the diagonalized matrix are positive and others are negative.

Their signs determine the direction of information travel. According to the sign of these eigenvalues, A_i and B_i can be written as

$$A_i = A_{i+} + A_{i-}$$
$$B_i = B_{i+} + B_{i-}$$

where A_{i+} and B_{i+} include only positive diagonal elements and A_{i-} and B_{i-} include the negative ones. Thus, the original matrices are split as

$$A_{i+} = S_x^{-1}\Lambda_{A+}S_x$$
$$A_{i-} = S_x^{-1}\Lambda_{A-}S_x$$
$$B_{i+} = S_y^{-1}\Lambda_{B+}S_y$$
$$B_{i-} = S_y^{-1}\Lambda_{B-}S_y .$$

As a result, the fluxes passing through a cell surface, for instance, at i+1/2,j, can be evaluated as follows (see Fig.4.2),

$$(F_i)_{i+1/2,j} = A_{i+}U_{i,j} + A_{i-}U_{i+1,j}$$
$$(G_i)_{i+1/2,j} = B_{i+}U_{i,j} + B_{i-}U_{i+1,j} .$$

Considering the flux vector splitting technique, the hybrid

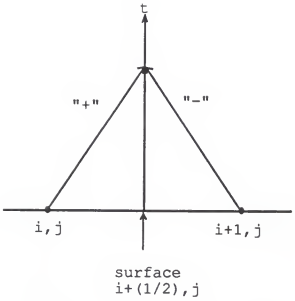


Figure 4.2 Direction of information travel in time to surface i+1/2,j.

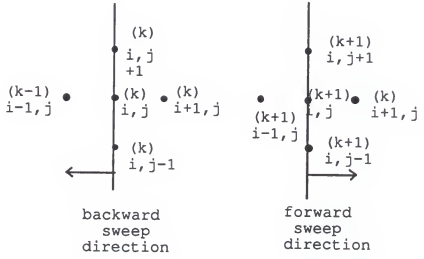


Figure 4.3 Sweep direction for line Gauss-Seidel iteration.

implicit-explicit algorithm is finally summarized as

Predictor:

$$\Delta U^{n}_{i,j} = -\Delta t \left[\frac{D_+ F_i}{\Delta z} + \frac{D_+ (G_i - G_v)}{\Delta x} - H \right]^{n}_{i,j}$$

$$\left[I + \Delta t \left(\frac{D_+ \cdot A_-}{\Delta z} + \frac{D_- \cdot A_+}{\Delta z} \right) + \Delta t \left(\frac{D_+ \cdot B_-}{\Delta x} + \frac{D_- \cdot B_+}{\Delta x} \right) \right. \quad (4.8)$$

$$\left. - \frac{\Delta t}{\delta x} (\delta G_v) \right] \delta U^{\overline{n+1}}_{i,j} = \Delta U^{n}_{i,j}$$

$$U^{\overline{n+1}}_{i,j} = U^{n}_{i,j} + \delta U^{\overline{n+1}}_{i,j}$$

Corrector:

$$\Delta U^{\overline{n+1}}_{i,j} = -\Delta t \left[\frac{D_- \cdot F_i}{\Delta z} + \frac{D_- \cdot (G_i - G_v)}{\Delta x} - H \right]^{\overline{n+1}}_{i,j}$$

$$\left[I + \Delta t \left(\frac{D_+ \cdot A_-}{\Delta z} + \frac{D_- \cdot A_+}{\Delta z} \right) + \Delta t \left(\frac{D_+ \cdot B_-}{\Delta x} + \frac{D_- \cdot B_+}{\Delta x} \right) \right. \quad (4.9)$$

$$\left. - \frac{\Delta t}{\delta x} (\delta G_v) \right] \delta U^{\overline{n+1}}_{i,j} = \Delta U^{\overline{n+1}}_{i,j} + \Delta S^{n}_{i,j}$$

$$U^{n+1}_{i,j} = \frac{1}{2} \left(U^{n}_{i,j} + U^{\overline{n+1}}_{i,j} + \delta U^{n+1}_{i,j} \right)$$

where

(1) the superscripts n, $\overline{n+1}$, and n+1 refer to present, predicted, and new solution values,

(2) the subscripts i and j refer to a special set of mesh point locations, ($i\Delta x$, $j\Delta y$),

(3) ΔU is the temporal change in the solution during

time interval Δt and is evaluated by a local explicit finite difference approximation,

(4) the difference operators, $D_+\cdot$, $D_-\cdot$ and $\bar{\delta}$, represent forward, backward, and central difference approximations applied to all factors to the right of the operator with respect to the direction indicated in the denominator of the difference operator quotient, i.e.,

$\left\{\frac{D_+ \cdot A_-}{\Delta Z} + \cdots\right\} \delta U_{i,j}$ is equivalent to

$$\frac{A_{-i+1,j}\delta U_{i+1,j} - A_{-i,j}\delta U_{i,j}}{\Delta z} \cdots .$$

The implicit equation of each step can be written as

$$B\delta U^{n+1}_{i,j+1} + A\delta U^{n+1}_{i,j} + C\delta U^{n+1}_{i,j-1} + D\delta U^{n+1}_{i+1,j} + E\delta U^{n+1}_{i-1,j} = \Delta U^{n}_{i,j} \tag{4.10}$$

where the coefficients A, B, C, D, and E are 4 x 4 block matrices. Equation (4.10) can be solved by the line Gauss-Seidel iterative procedure [37,38]. The stencil of mesh points used is shown in Figure 4.3. A line at a time is solved for simultaneously by using the latest available values ahead of and behind the line. The parameter k is the iteration index. The relaxation algorithm is implemented by

alternately sweeping in the backward and forward streamwise directions. Alternating the direction of the implicit line inversion has been proven to be beneficial to the solution of the Navier-Stokes equations with highly stretched grids. It is also consistent with the nature of transonic and viscous separated flow [39].

4.3 Boundary and Initial Conditions

For finite volume schemes, the reflection condition has often been used to treat the flow on a wall boundary. The boundary conditions are summarized as

$$
\begin{aligned}
U_{i,1} &= \begin{Bmatrix} U_{i,2} & slip \\ -U_{i,2} & no\ slip \\ 2U_{wall} - U_{i,2} & moving\ wall \end{Bmatrix} \\
V_{i,1} &= \begin{Bmatrix} -V_{i,2} & no\ slip \\ 2V_{wall} - V_{i,2} & moving\ wall \end{Bmatrix} \\
T_{i,1} &= \begin{Bmatrix} T_{i,2} & adiabatic \\ 2T_{wall} - T_{i,2} & isothermal \end{Bmatrix}
\end{aligned}
\tag{4.11}
$$

where U and V are the velocities of the z and r directions, respectively. The constant stagnation pressures and temperatures at the inflow and outflow boundary are used. Initially, the flow is at rest with pressure and temperature set everywhere to their total values at the entrance. The following medium starts to move by setting the exit pressure to a lower pressure than the inlet pressure.

4.4 Convergence Criterion

Since an iterative solution procedure is used to solve the equations, it is necessary to establish a convergence criterion which measures the degree to which a computed solution satisfies the finite difference equation. In the present work, this convergence criterion is based on the value of velocity residual defined by

$$R = \frac{\sum_{i=1}^{imax} \sum_{j=1}^{jmax} \{[u_{i,j}^{n} - u_{i,j}^{n-1}]^2 + [v_{i,j}^{n} - v_{i,j}^{n-1}]^2\}^{1/2}}{(imax)\,(jmax)} \qquad (4.12)$$

When the reduction of the velocity residual is saturated to an asymptotic value, the solution is considered converged. In general, the asymptotic value of the residual varies according to the flow conditions and the geometries used, and the values obtained in this study were mostly less than the value of $R = 3 \times 10^{-3}$. The convergence was also able to be graphically determined.

CHAPTER 5
EVALUATION OF HEAT TRANSFER RATES

5.1 Convective Heat Transfer Rate

There are two ways to determine the convective heat flux into the wall boundary. One way is to use Fourier's law as the product of the fluid conductivity and temperature gradient near the wall. The other way is to use the convective heat transfer coefficients which can be obtained from the literature for fully developed flows of various types in conduits or in simple geometries. For the first method, a very fine grid near the wall is required so that the viscous sublayer can be properly resolved and then the heat flux is directly determined as

$$q_c'' = -k_c \frac{\partial T}{\partial r}\Big|_w \qquad (5.1)$$

where the thermal conductivity, k_c, is evaluated at the average temperature of the cell to the wall. The temperature gradient in equation (5.1) is numerically calculated using the estimated cell temperature and the wall temperature. Usually the convective heat transfer is predicted with the convective heat transfer coefficient, h_c,

as defined by

$$h_c = - \frac{k_c \frac{\partial T}{\partial r}|_w}{T_w - T_b} \tag{5.2}$$

where T_w is the wall temperature and T_b is the bulk temperature. The bulk temperature is averaged over the flow cross-sectional area according to the following relation

$$T_b = \frac{\int_A \rho C_p u T dA}{\int_A \rho C_p u dA} \tag{5.3}$$

where A is the cross-sectional flow area, C_p is the specific heat at constant pressure, ρ is the density, and u is the axial flow velocity. On the other hand, the empirical correlations can be used to calculate the convective heat transfer coefficients. A typical correlation for the wall convective heat transfer coefficient for fully developed turbulent flow in pipes is due to Dittus and Boelter [2]

$$N = 0.023\, Re^{0.8} Pr^{1/3} \qquad T_b > T_w \tag{5.4}$$

where the Nusselt number is defined as hD/k_c. The Nusselt number obtained with the above equation (5.4) is compared with the result predicted by the near-wall diffusion method.

5.2 Radiative Heat Transfer Rate

Radiative heat flux is also predicted by two different approaches in this study. From the diffusion approximation, the radiative flux is estimated by the temperature gradient from the wall to the adjacent cell and the radiative conductivity defined in Section 2.2.1. The expression is written as

$$q_r'' = -k_r \frac{\partial T}{\partial r}\Big|_w \tag{5.5}$$

where the temperature gradient at the wall is numerically calculated in the same manner as in the previous section. For the approximation when using the one-dimensional equation of radiative transfer, the radiative flux into the wall is the net radiative heat flux resulting from the exchange between the fluxes which come from the inner gas layer within 5 mean free path (mfp) and the fluxes emitted from the wall. The net radiative flux at the wall is calculated by

$$q_w'' = \sum_j^2 q_{j\text{-}wall}'' - \sigma T_w^4 \tag{5.6}$$

where the subscript w indicates the wall.

CHAPTER 6
RESULTS AND DISCUSSION

In this chapter the computational results for the fluid flow and heat transfer are presented. This chapter is composed of two major parts, the first of which describes the results for the flow in both a straight tube and a tube with a circular arc bump. The straight tube is utilized to verify the fluid flow and heat transfer models used in this study. The tube with a circular arc bump is used to test the model's capability of capturing shock waves. The second part presents the numerical results for converging-diverging nozzle flows. Air and UF_4 gas are the fluids considered in this study. Air is used to compare the numerical results with existing experimental or analytical data, and UF_4 is used to simulate the flow of a UTVR-MHD system.

6.1 Calculations in a Straight Tube and a Tube with a Circular Arc Bump

6.1.1 Verification of Modelling

The developed model is validated in this section. Two simple geometrical configurations, a straight tube and a tube with a circular arc bump on the inner wall, are used to verify the present model. The straight tube has a diameter-

to-length (L/D) ratio of 60, and a 98 x 60 grid is used. The velocity distributions within fully-developed regions are presented for Reynolds numbers within a range of $1.6 \times 10^4 \leq Re \leq 3.9 \times 10^5$. These numerical results are compared to the velocity profile depicted by the 1/7th power law in Figure 6.1. The velocity distribution of the 1/7th power law represents the fully-developed turbulent flow distribution [24]. As shown in Figure 6.1, the results are in reasonable agreement. The comparison is, however, not suited for a wide range of Reynolds numbers.

Therefore, the universal velocity distribution, which represents the velocity distribution for fully-developed turbulent tube flows in a wide range of Reynolds numbers, is used for the verification of turbulent flow solution. The numerical results are compared to Laufer's and Deissler's experimental data [24] for the mean velocity in a fully-developed isothermal tube flow as shown in Figure 6.2. The excellent agreement between the present model and the experimental data indicates the validity of the developed model and the related assumptions for prediction of turbulent flows.

In addition, a close-up view of the streamwise direction velocity profiles at various locations downstream are presented in Figure 6.3. The temperature profiles, which are obtained at a constant wall temperature of 1000 K

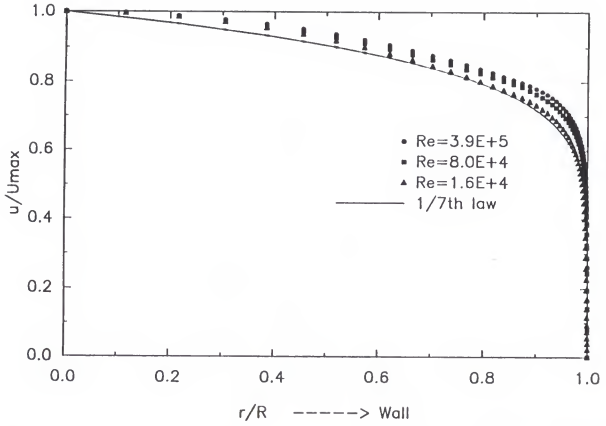


Figure 6.1 Calculated velocity distribution in a straight tube for varying Reynolds number and velocity distribution following 1/7th law.

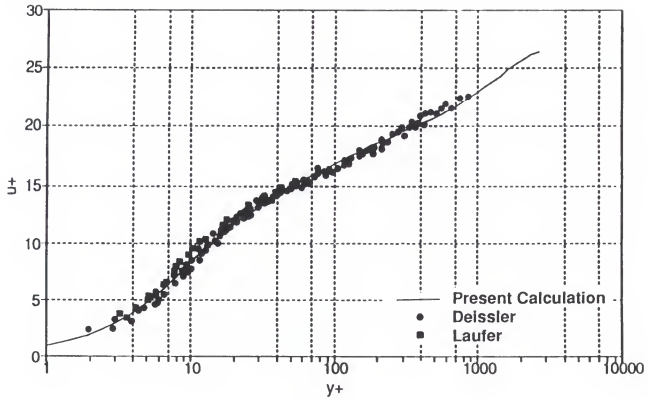


Figure 6.2 Velocity distribution for turbulent, isothermal flow in tubes.

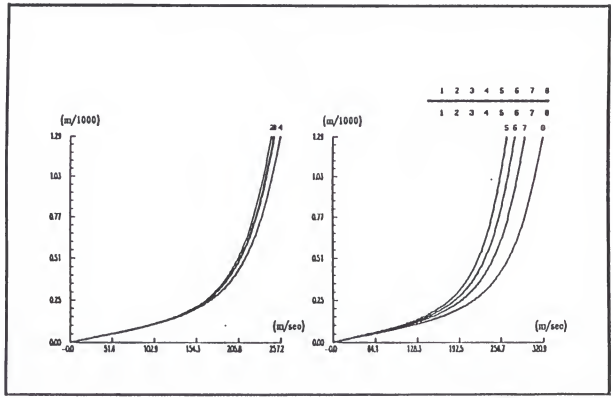


Figure 6.3 Close-up view of a typical near-wall velocity profile as calculated for an isothermal situation.

and an inlet stagnation temperature of 1500 K, are presented in Figure 6.4. For both results, parabolic profiles of boundary layers are observed. In Figure 6.5, the pressure drop along the axis is shown. The pressure drop is presented as the ratio of static to stagnation pressure.

All of the above results are obtained from the solution of the flows at the steady state. In order to verify the steady state solution, the convergence history for the velocity residual is examined, as shown in Figure 6.6. The steady state is reached when the velocity residual approaches an asymptotic value. In this test case, the steady state is reached in around 80 iterations with a residual of about 4×10^{-3}.

Another geometry of interest to this study is a converging-diverging nozzle in which transonic flow and shock waves occur. The problem of flow in a channel with an inward circular arc bump is chosen to evaluate the code for subsonic and transonic steady-state modeling. This particular problem is well suited for testing of solutions of the subsonic and transonic flow. The geometry and the grids are easy to generate accurately, and the symmetry and geometrical simplicity aid in the interpretation of the results.

For subsonic and transonic modeling, the geometrical configuration is shown in Figure 6.7, and the thickness of the bump is 10% of the radius of the channel. The

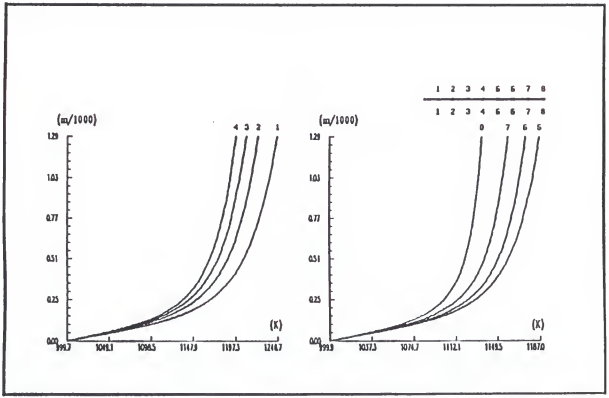


Figure 6.4 Close-up view of a typical near-wall temperature profile as calculated for an isothermal situation.

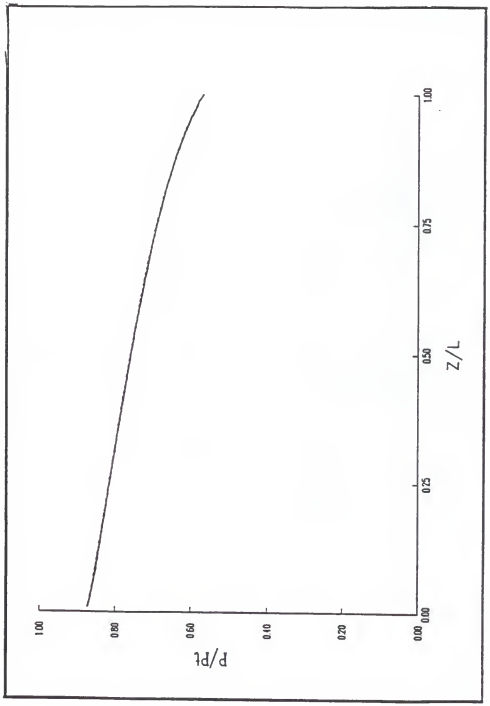


Figure 6.5 Calculated surface pressure distribution along the tube.

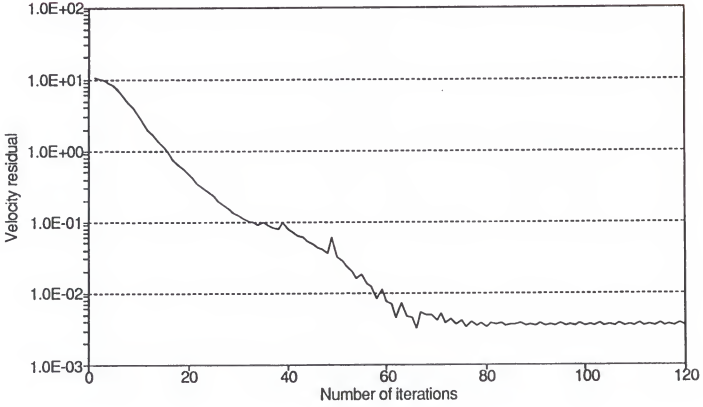


Figure 6.6 Convergence rate for the steady state solution in a straight tube.

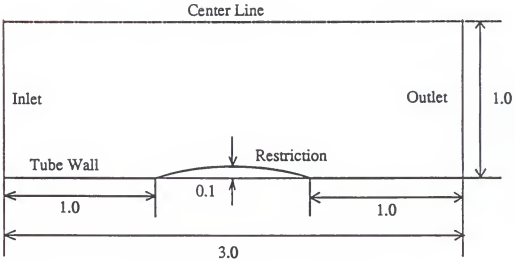


Figure 6.7 Geometry of a tube with a circular restriction.

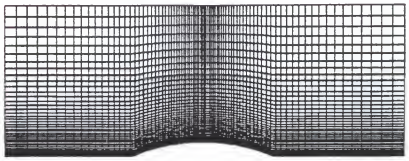


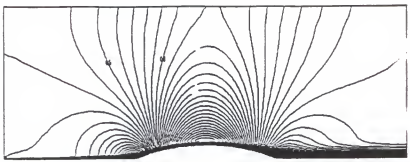
Figure 6.8 A 90x60 grid system.

computational mesh, composed of 90 x 60 grid lines as shown in Figure 6.8, is used in both subsonic and transonic calculations. With the geometrical configuration mentioned above, the boundary conditions for subsonic flow are as follow. Solid-wall boundary and symmetric boundary conditions are applied to the solid walls and the center line. The inflow boundary is on the left side. The stagnation pressure at the inlet is 1 atm, while the back pressure is kept at 0.8 atm. The inlet temperature is maintained at 2500 K, and the wall temperature is kept at 1000 K.

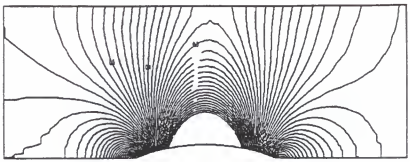
Figure 6.9 displays the isomach lines, the normalized pressure, and the normalized temperature lines at steady state. As depicted in the Figure 6.9, the numerical solutions are quite symmetric about the midchord, which is a good indication of the accuracy of the solution for this subsonic application. The surface pressure distribution along the inner tube wall plotted in Figure 6.10 presents the symmetric characteristic quite well.

For the case of transonic flow, the same flow conditions used in the subsonic flow are applied but the back pressure is changed to 0.25 atm. From the calculation, a supersonic region appears in the solution which is depicted by a shock as shown in Figure 6.11.

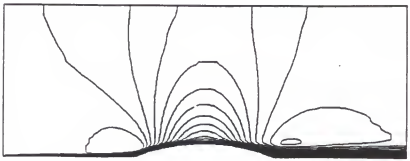
The transient isomach contours, recorded at t=0.0003, 0.0008, 0.0013, and 0.0022 seconds, respectively, are shown in



(a)



(b)



(c)

Figure 6.9 Computed contours at steady state for subsonic flow. (T_{in}=2500 K, T_w=1000 K, P_{in}=1 atm, P_{out}=0.8 atm, and Pr=0.72) (a) Mach number (b) normalized pressure (c) normalized temperature

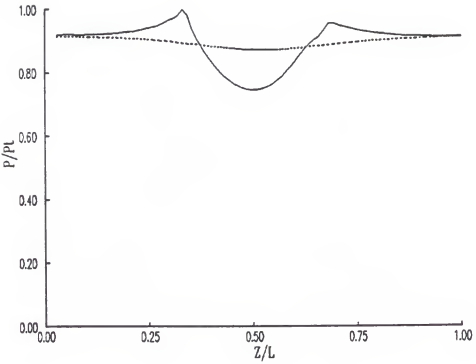


Figure 6.10 Computed pressure distributions for a subsonic flow regime; surface pressure (solid line) and centerline pressure (dotted line) (T_{in}=2500 K, T_w=1000 K, P_{in}=1 atm, P_{out}=0.8 atm, and Pr=0.72).

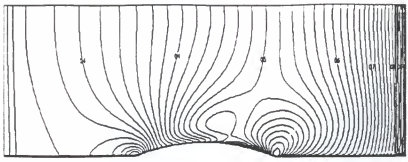
Figure 6.11. The isomach contour of the steady-state solution shows a clear sharp shock wave which is located around the trailing edge of the bump.

The normalized pressure and temperature contours are also presented in Figures 6.12 and 6.13, respectively. The same shapes of shock formations are observed. In addition, the clear discontinuity due to the shock waves is observed in the surface pressure distribution, as shown in Figure 6.14.

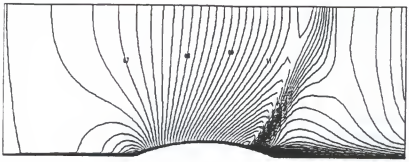
6.1.2 Effect of Boundary Cell Size in Convective Heat Transfer Rate

As previously mentioned, convective heat transfer at the wall occurs by molecular diffusion and is determined by Fourier's law. When using Fourier's law to determine the convective wall heat flux, the size of the boundary grids is of crucial importance because variation in the size of the boundary grids significantly influences the calculation of the temperature gradient which is directly proportional to the wall heat flux.

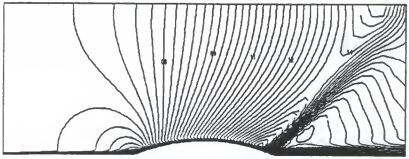
A mesh refinement study is conducted to determine the largest grid size below which the computed wall heat transfer rate does not introduce a significant error. Figure 6.15 presents the convective heat transfer rates estimated as a function of the size of the boundary grid, dy, in a 60 x 60 grid system. The dy is the distance from the wall to the point where the radial temperature gradient



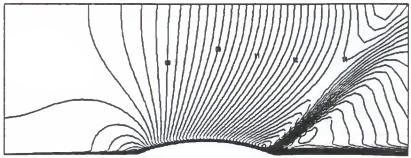
(a)



(b)

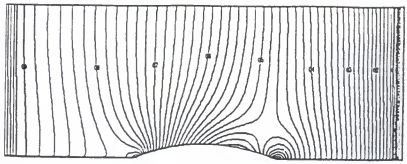


(c)

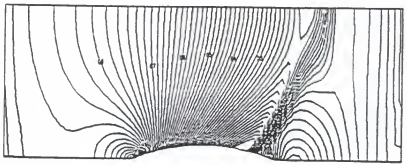


(d)

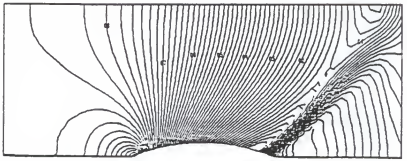
Figure 6.11 Calculated isomach number lines at different time for transonic flow. (T_{in}=2500 K, T_v=1000 K, P_{in}=1 atm, P_{out}=0.25 atm, and Pr=0.72) (a) t=0.0003 sec. (b) t=0.0008 sec. (c) t=0.0013 sec. (d) t=0.0022 sec.



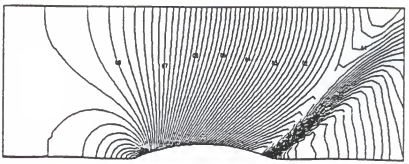
(a)



(b)



(c)



(d)

Figure 6.12 Calculated normalized pressure contours at different time for transonic flow. (T_{in}=2500 K, T_w=1000 K, P_{in}=1 atm, P_{out}=0.25 atm, and Pr=0.72) (a) t=0.0003 sec. (b) t=0.0008 sec. (c) t=0.0013 sec. (d) t=0.0022 sec.

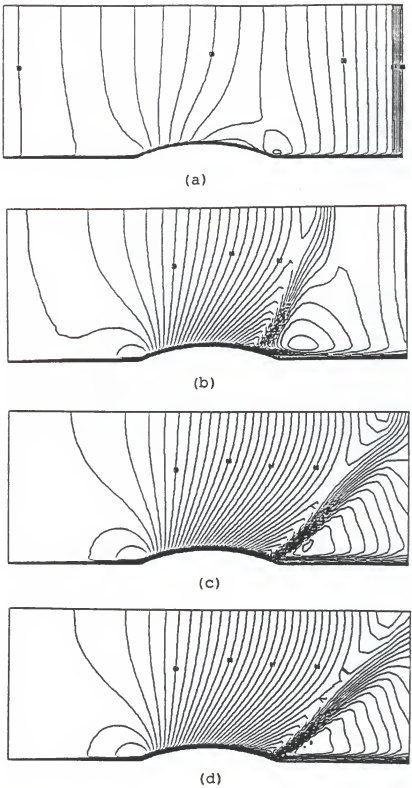


Figure 6.13 Calculated normalized temperature contours at different time for transonic flow. (T_{in}=2500 K, T_w=1000 K, P_{in}=1 atm, P_{out}=0.25 atm, and Pr=0.72) (a) t=0.0003 sec. (b) t=0.0008 sec. (c) t=0.0013 sec. (d) t=0.0022 sec.

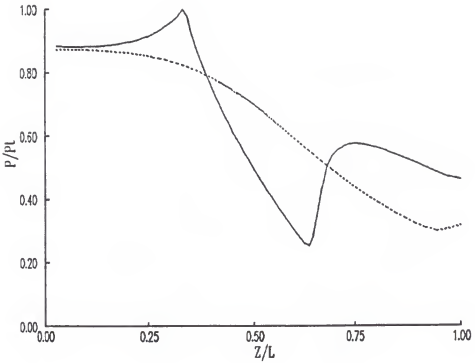


Figure 6.14 Calculated pressure distributions for a transonic flow regime; surface pressure (solid line) and centerline pressure (dotted line) (T_{in}=2500 K, T_w=1000 K, P_{in}=1 atm, P_{out}=0.25 atm, and Pr=0.72).

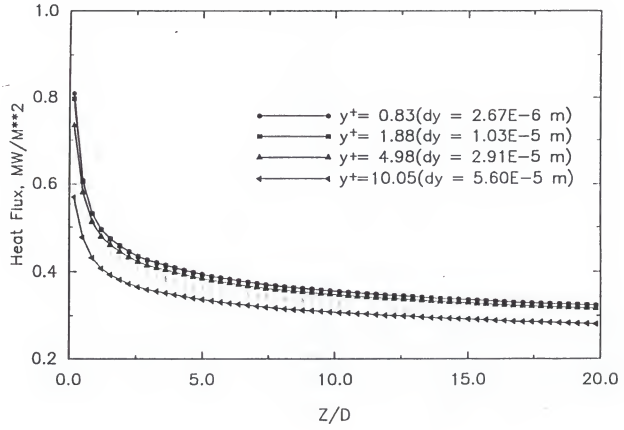


Figure 6.15 Heat transfer rates obtained by Navier-Stokes solver for various boundary cell sizes. A 60x60 grid is used. (T_{in}=4000 K, T_w=1800 K, P_{in}=1 atm, and P_{out}=0.5 atm)

at the wall is calculated. More detail of the grid sizes near the wall is given in Table 6.1. In this calculation, a straight tube, 5 cm in diameter and 1 m in length, is used. The initial inlet temperature is 4000 K, and the wall temperature is kept at 1800 K. The stagnation pressure at the inlet is 1 atm, while the back pressure is 0.5 atm.

For a grid size, dy, greater than 2.91×10^{-5} m, the computed heat flux is underestimated, as is shown in Figure 6.15. For the grid sizes of dy = 2.67×10^{-6} and dy = 1.03×10^{-5}, the calculated heat transfer rates are equal. In terms of the dimensionless distance from the wall, $y^+ = yv^*/\nu$, where v^* is the frictional velocity and ν is the kinematic velocity, the maximum allowable boundary grid size is approximately 5. This indicates that accurate prediction of the temperature gradient near the wall requires grid sizes less than the thickness of the laminar sublayer. Note that the dimensionless thickness, y^+, of the laminar sublayer in turbulent flow is 5 [40]. In this case, as shown in Table 6.1, seven node points are located within the laminar sublayer, $y^+ = 5$.

A grid sensitivity analysis is performed to confirm the relation between the thickness of the laminar sublayer and the maximum allowable grid size near the wall. The thickness of the laminar sublayer is varied with the use of different flow conditions, such as system pressures or inlet

Table 6.1 Details of grid sizes near wall for a tube in 5 cm diameter and 1 m length. 60 x 60 grid system. T_{in}=4000 K, T_{wall}=1800 K, P_{in}=1 atm, and P_{out}= 0.5 atm.

Grid # from Wall	Actual Distance from Wall in m (x 10^{+5})	y^+ $(=y\frac{\sqrt{\tau_0/\rho}}{\nu})$
1	0.266	0.83
2	0.693	1.32
3	1.035	1.88
4	1.421	2.52
5	1.858	3.24
6	2.353	4.06
7	2.914	4.98
8	3.548	6.02
9	4.266	7.20
10	5.079	8.54
11	5.998	10.05
12	7.040	11.77

stagnation temperatures. For instance, higher inlet stagnation pressure and back pressure are chosen to calculate the convective wall heat flux while other conditions are kept the same.

Figure 6.16 shows the convective wall heat transfer rates which are estimated for different choices of grid size. The inlet and back pressures in this case are 10 and 9.5 atm, respectively, and the grid size is 60 x 80. However, the inlet and exit temperatures are kept the same as those in the 60 x 60 grid case.

In this case the estimated heat fluxes are more sensitive to the grid size, as shown in Figure 6.16. Therefore, it is evident from Figure 6.16 that if the first grid from the wall is located in the laminar sublayer, the estimated convective heat transfer rate is accurate and reasonable. Table 6.2 lists the calculated grid sizes of a 60 x 80 grid system. According to the data given in Table 6.2, two grid points are located within the laminar sublayer, even though more grids in the radial direction are allocated. This indicates that the thickness of the viscous sublayer is thinner for the high pressure case where the higher density increases the value of Reynolds numbers.

From this analysis, it is evident that accurate convective heat transfer rates can be estimated if the boundary cell size in the radial direction is smaller than the laminar sublayer, $y^+ \leq 5$, thickness. The use of a boundary

Table 6.2 Details of grid sizes near wall for a tube in 5 cm diameter and 1 m length. 60 x 80 grid system. T_{in}=4000 K, T_{wall}=1800 K, P_{in}=10 atm, and P_{out}=9.5.

Grid # from Wall	Actual Distance from Wall in m (x 10^{+5})	$y^+ \left(=y\frac{\sqrt{\tau_0/\rho}}{\nu}\right)$
1	0.125	1.65
2	0.386	3.46
3	0.672	5.43
4	0.983	7.58
5	1.323	9.92
6	1.694	12.48
7	2.098	15.28
8	2.540	18.32
9	3.022	21.65
10	3.548	25.28
11	4.122	29.25
12	4.749	33.57

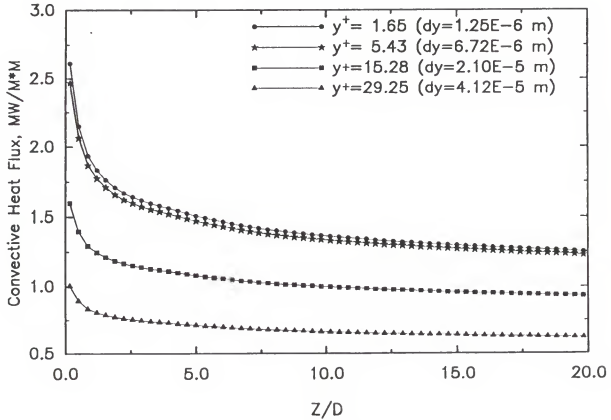


Figure 6.16 Heat transfer rates obtained by Navier-Stokes solver for various boundary cell sizes. A 60x80 grid is used. (T_{in}=4000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

cell size larger than the laminar sublayer thickness results in an underestimate of the calculated wall heat flux.

Using the present heat transfer model, the calculated convective wall heat transfer rate is compared with existing experimental data for a developing, isothermal pipe flow.

The result is presented in Figure 6.17 and compared with Boelter's experimental data [12] for the air flow at a Reynolds number of 53000 at the exit. The agreement between the calculated result and the experimental data is reasonably good. Moreover, these results agree better beyond the diameter-to-length ratio (L/D) of about 10.

Figure 6.18 presents the results of the comparison between the directly calculated Nusselt number from the transverse temperature gradient of the thermal field solution and the Nusselt number evaluated using the Dittus-Boelter correlation [2]. For this test calculation, the stagnation inlet and the back pressure are 10 atm and 9.5 atm, respectively, and the inlet temperature is 4000 K and the wall temperature is maintained at 1800 K. Results are in reasonably good agreement as the flow becomes fully developed. From this analysis, it is concluded that the convective wall heat transfer rate is accurately predicted by the developed fluid flow and heat transfer model.

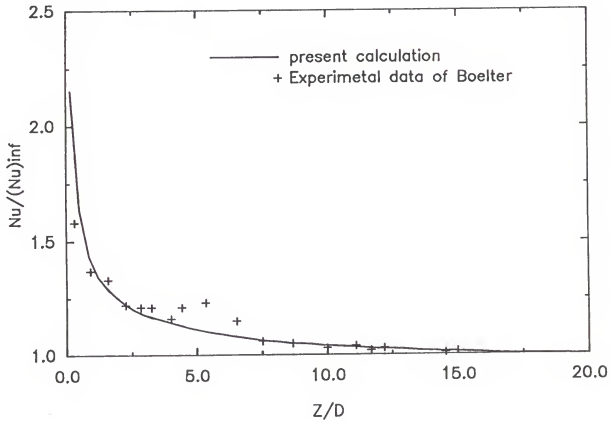


Figure 6.17 Nusselt number versus axial position for a developing isothermal pipe flow at a Reynolds number of 53000. $(Nu)_{inf}$ is the Nusselt number evaluated at Z/D=20.

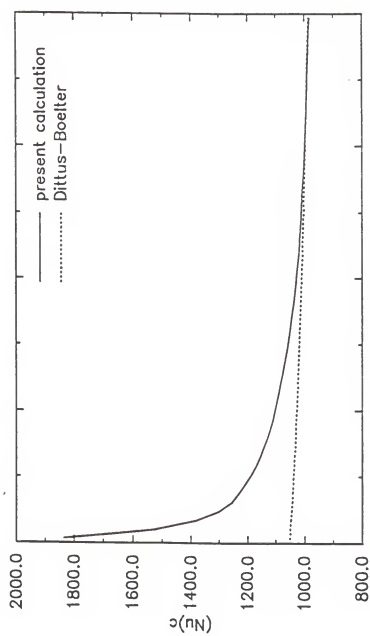


Figure 6.18 Comparison of Nusselt number as obtained from the computed heat flux with the result obtained by using the Dittus-Boelter correlation. A 60x60 grid is used. (T_{in}=4000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

6.1.3 Evaluation of Diffusion Approximation

An assessment is made of the diffusion approximation as a matter of interest. The local radiative heat flux at various tube wall locations is computed by two different approximations, the diffusion and the simplified integral approximation. A comparison is made using the same tube dimension as that used in the previous section with a 60 x 60 grid. The inlet temperature is taken to be 4000 K while the tube wall temperature is assumed to be 1800 K.

As previously mentioned, the Rosseland's diffusion approximation, which can be applicable for a certain range of absorptivities, is used to account for the radiative heat transfer. The limiting range of the absorptivity for the diffusion model is examined by comparing the radiative heat flux estimated by the two different approximations mentioned above.

The comparative results are plotted in Figure 6.19 which presents the ratio of the heat flux of the diffusion model to that of the integral model at various axial positions. This comparison is performed by varying the absorptivity with a change of flow field pressure. The mean absorptivities, based on a tube wall temperature of 1800 K and inlet pressures ranging from 10 atm to 100 atm, vary from a value of 111.1 cm^{-1} to a value of 1111.1 cm^{-1}. As seen in Figure 6.19, the diffusion approximation consistently underestimates the heat flux by about 30%

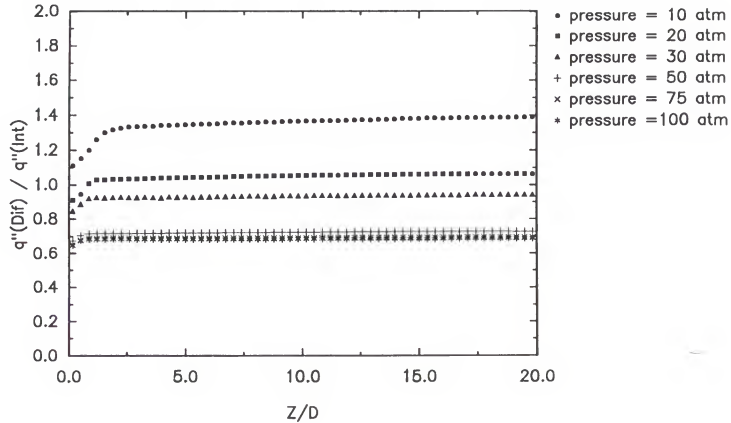


Figure 6.19 Comparative result for radiative heat flux between diffusion approximation and 1-D integral approximation for varying gas opacity due to different flow conditions.

compared to the results of the integral approximation for mean absorptivities greater than 555.5 cm^{-1}. Moreover, it is found that the discrepancy between the diffusion approximation and the simplified integral approximation becomes more significant when applied to an optically thin gas, such as a gas with an absorptivity of less than about 100 cm^{-1}. Therefore, it is suggested that the diffusion approximation should not be used for an optically thin gas.

6.1.4 Convective and Radiative Heat Transfer

The heat fluxes obtained with the developed models are presented for both convective and radiative heat transfer. The convective and radiative heat transfer rates presented in this section are calculated by using the near-wall diffusion method and the 1-D integral approximation, respectively. The prediction of the radiative heat transfer including the effect of the tube wall temperature and the gas inlet temperature are presented here. In order to examine the wall temperature effect, the tube wall temperatures of choice are 1800, 2100, and 2400 K, while the inlet temperature is kept at 4000 K.

The heat fluxes obtained with the different wall temperatures are plotted in Figure 6.20. It is seen that the radiative fluxes increase as higher wall temperatures are selected. This behavior is due simply to the fact that the radiative wall heat flux is dependent on the local

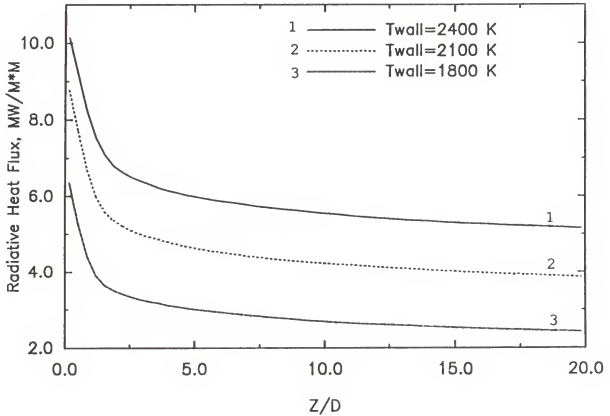


Figure 6.20 Radiative heat flux calculated for varying wall temperatures. (T_{in}=4000 K, P_{in}=10 atm, and P_{out}=9.5 atm)

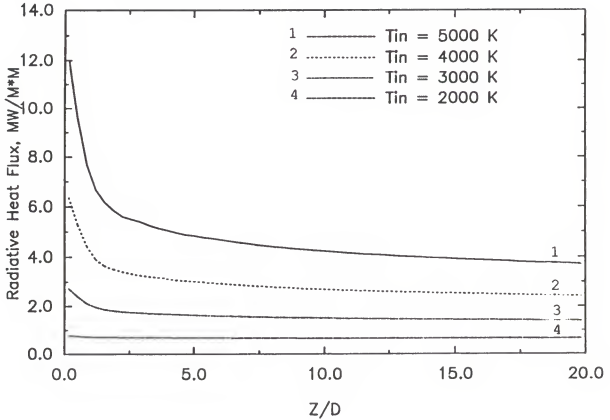


Figure 6.21 Radiative heat flux obtained for different inlet stagnation temperatures. (T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

conditions at the wall as well as the gas inlet temperatures. This result is displayed in Figure 6.21. If the gas inlet temperatures are high, the high radiative fluxes occur at the wall, as expected, but the effect of changing the inlet gas temperature is less than that of changing the wall temperature. This indicates that the radiative heat transfer at the wall is more likely to be affected by the local conditions at the wall.

Using the developed model, the heat fluxes due to the combined radiative and convective heat transfer for the five cases summarized in Table 6.3 are presented. Figures 6.22

Table 6.3 Flow conditions for calculation and comparison of radiative and convective heat transfer rates.

	P_{in} (atm)	P_{out} (atm)	T_{in} (K)	T_w (K)
CASE 1	10	9.5	4000	1800
CASE 2	10	9.5	3000	1800
CASE 3	10	9.5	4000	2100
CASE 4	30	29.5	4000	1800
CASE 5	50	49.5	4000	1800

and 6.23 display the results of the calculations for cases 1 and 2 that differ only in the gas inlet temperature which also changes the value of the mean opacity of the gas. The radiative flux in Figure 6.22, for an inlet temperature of

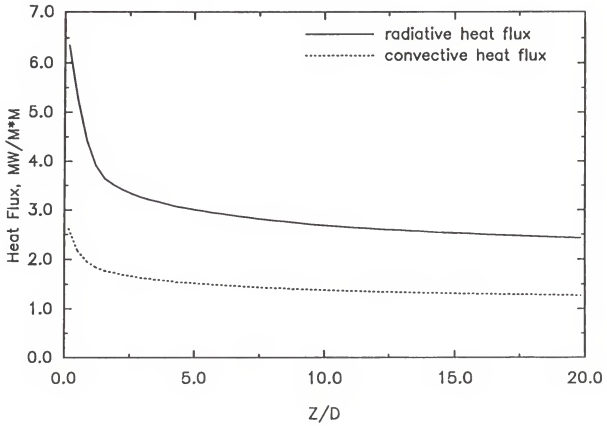


Figure 6.22 Convective and radiative heat fluxes for flow conditions. (T_{in}=4000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

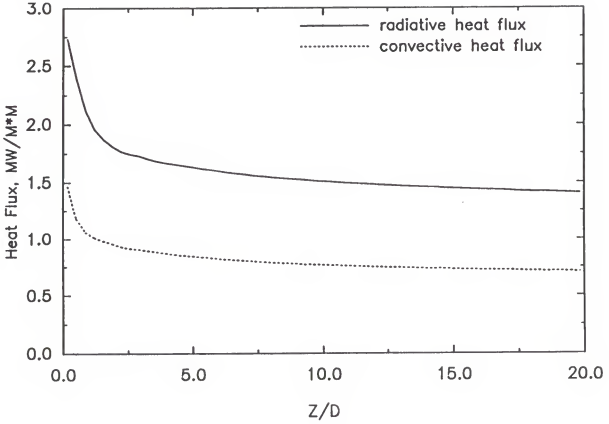


Figure 6.23 Convective and radiative heat fluxes for flow conditions. (T_{in}=3000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

4000 K, is almost two times larger than the convective flux, while the radiative flux for the inlet temperature of 3000 K is slightly less than two times in the magnitude of the convective flux. Both radiative and convective fluxes for case 1 are almost twice as large as those of case 2 where the inlet gas temperature is reduced by 1000 K.

Results of radiative and convective heat transfer calculations for case 3 are presented in Figure 6.24. Flow conditions in case 3 are similar to those of case 1, with the exception that the wall temperature is raised by 300 K. It is seen that the radiative flux is about 60% greater than that of case 1, while the convective flux remains almost the same.

Finally, with the change of the flow field pressure, the radiative and convective heat fluxes are estimated and plotted in Figures 6.25 and 6.26. The flow conditions for cases 4 and 5 are similar to those of case 1, but the flow field pressure is increased by factors of 3 and 5. As can be seen in the figures mentioned above, the convective heat flux increases, but the radiative heat flux decreases as the flow field pressure becomes higher and higher. The increase of the convective heat flux seems to follow the change of the density which is due to the pressure increase. The decrease in the magnitude of the radiative flux may be

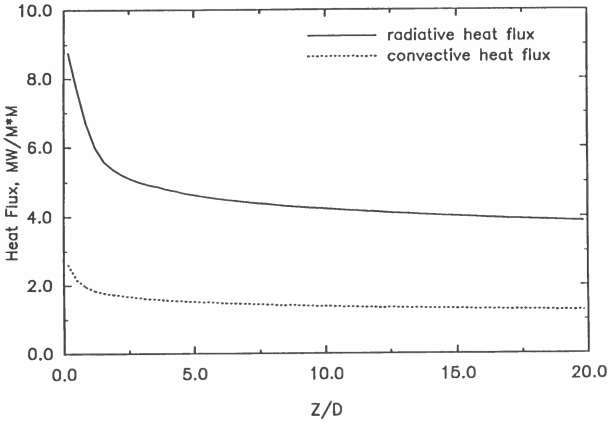


Figure 6.24 Convective and radiative heat fluxes for flow conditions. (T_{in}=4000 K, T_w=2100 K, P_{in}=10 atm, and P_{out}=9.5 atm)

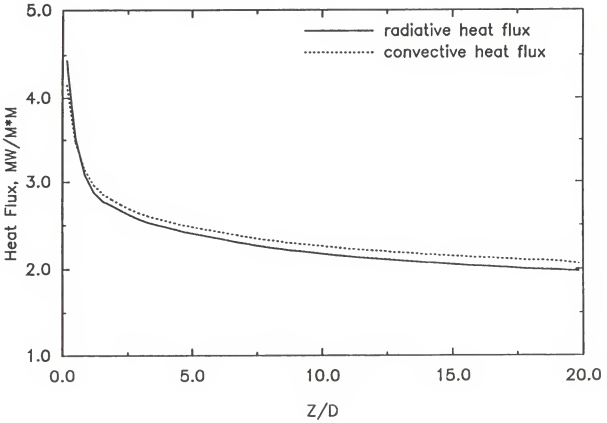


Figure 6.25 Convective and radiative heat fluxes for flow conditions. (T_{in}=4000 K, T_w=1800 K, P_{in}=30 atm, and P_{out}=29.5 atm)

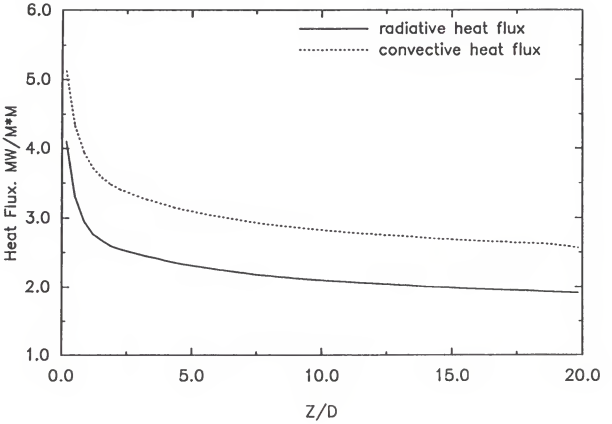


Figure 6.26 Convective and radiative heat fluxes for flow conditions. (T_{in}=4000 K, T_w=1800 K, P_{in}=50 atm, and P_{out}=49.5 atm)

simply caused by the shortening of the radiation mean free path by the higher molecular density. It is interesting to note that the magnitudes of the convective heat flux for cases 4 and 5 are greater than those of the radiative flux.

6.1.5 Effect of Internal Heat Generation in Heat Transfer Rate

UTVR-MHD systems are characterized by high internal heat generation rates in the gas. The developed model is used to predict the heat transfer rates for the flow of gases with internal heat generation. Uniform power densities ranging from 0 to 1000 w/cc are assumed and included in the energy equation as source terms.

The inlet gas temperature for the case under consideration is 2000 K which is a typical temperature at the core inlet of a UTVR-MHD system. The wall temperature is kept at 1800 K, and the inlet stagnation pressure is 10 atm. The back pressure is maintained at 9.5 atm.

First, the evolution of bulk temperature as a function of the tube length for different values of power density is shown in Figure 6.27. The increase of the bulk temperature at the power density of 1000 w/cc is about 2000 K. This indicates that a power density higher than 1000 w/cc is required to maintain the exit gas temperature of the tube at about 4000 K. The temperature of 4000 K is a typical temperature at the core exit required to achieve high efficiency in the MHD energy conversion system.

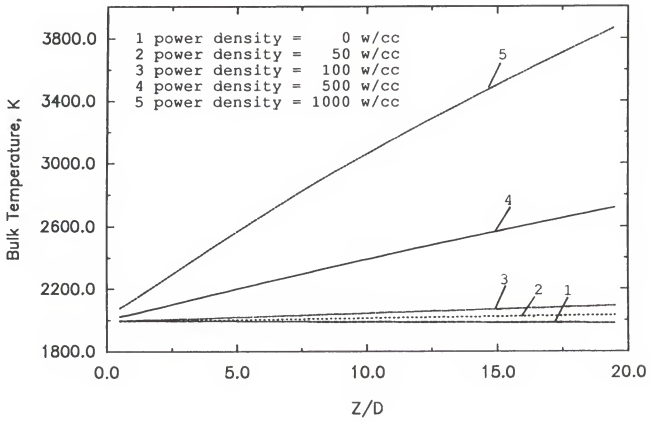


Figure 6.27 Bulk temperature evolution for power densities of 0, 50, 100, 500 w/cc, and 1000 w/cc. (T_{in}=2000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

Figure 6.28 presents the Nusselt numbers for the convective heat transfer at different power densities. The Nusselt numbers for power densities of 50 w/cc and 100 w/cc progressively increase in regions beyond the tube entrance. However, for high power densities such as 500 w/cc and 1000 w/cc, a drastic increase of Nusselt numbers occurs near the tube entrance.

These results indicate that for moderate power densities, 50 w/cc and 100 w/cc, the heat generated due to the internal heat generation and the heat removed by convection heat transfer in the axial direction are almost comparable for the positions, $L/D \leq 5$. As the flow proceeds downstream, the generated heat steadily exceeds the heat transferred through the wall. For very high power densities such as 500 w/cc and 1000 w/cc, the heat from the internal heat source considerably exceeds the transferred heat through the wall as the flow proceeds only a short distance into the tube.

Radiative heat transfer rates are estimated for power densities of 10 w/cc, 100 w/cc, and 500 w/cc, and the results are plotted in Figure 6.29. Since the radiative heat transfer is more strongly dependent on the gas temperature, the radiative heat transfer rate increases rapidly along the tube at a power density of 500 w/cc.

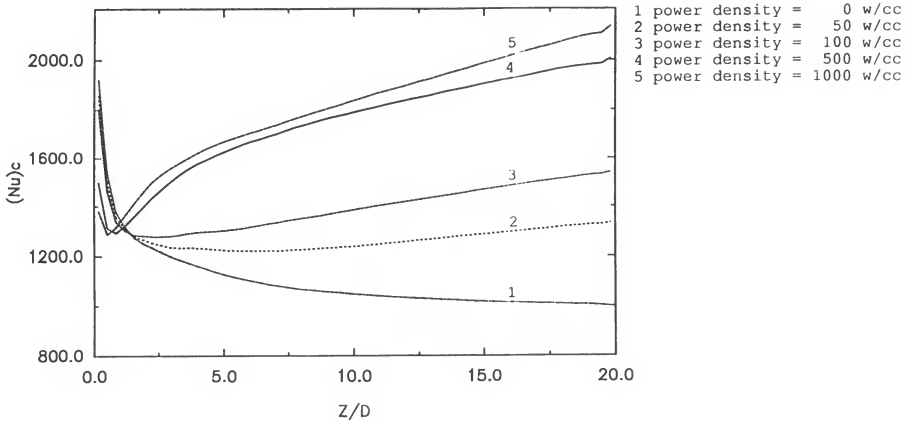


Figure 6.28 Convective Nusselt number computed for power densities of 0, 50, 100, 500 w/cc, and 1000 w/cc. (T_{in}=2000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

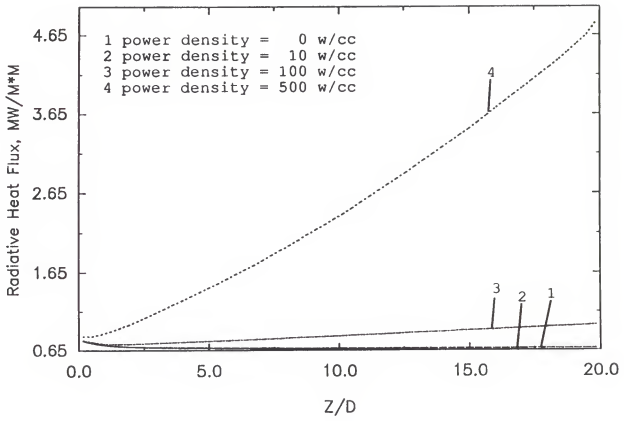


Figure 6.29 Radiative heat flux obtained for power densities of 0, 10, 100, and 500 w/cc. (T_{in}=2000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm)

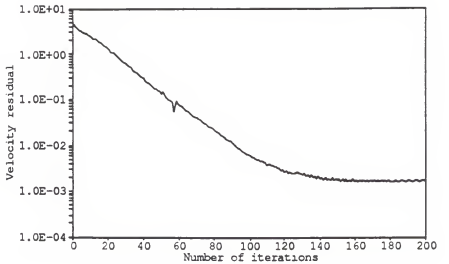
6.1.6 Test of Convergence

Since an iterative procedure is used to obtain a steady-state solution as a time asymptotic solution of the unsteady equations, a test of convergence is necessary. Thus, the convergence of the process is examined for different flow conditions. The velocity residuals as a function of the iteration numbers are plotted in Figure 6.30, and a steady state is reached when the velocity residual approaches an asymptotic value. As seen in these figures, the number of iterations for the steady state solutions is increased as pressure and temperature increase.

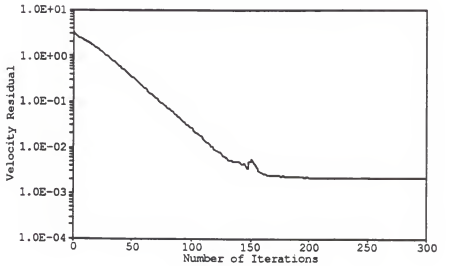
6.2 Calculations in a Converging-Diverging Nozzle

6.2.1 Solution of Flow and Thermal Field in a Nozzle

The axisymmetric Navier-Stokes code with the Baldwin-Lomax turbulent model is used to carry out a simulation of the converging-diverging nozzle in which the flow field is under the influence of large pressure gradients. With the thermal field solution obtained, convective and radiative heat transfer rates along the wall of the nozzle are predicted. Since the simplified integral approach, which is applicable only to a simple tube geometry, is inaccurate for the prediction of the radiative heat transfer for this nozzle geometry, the diffusion approximation is adopted.

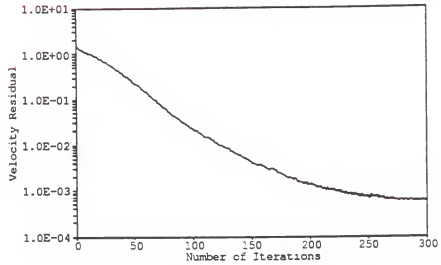


(a)

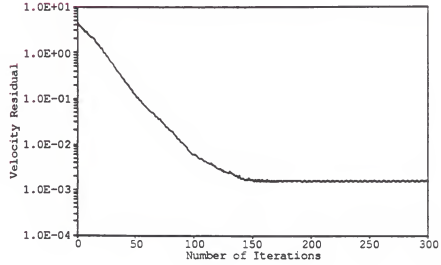


(b)

Figure 6.30 Convergence rates for different flow conditions.
(a) T_{in}=4000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm, Q=0
(b) T_{in}=5000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm, Q=0



(c)



(d)

Figure 6.30--continued.

(c) T_{in}=4000 K, T_w=1800 K, P_{in}=30 atm, and P_{out}=29.5 atm, Q=0

(d) T_{in}=4000 K, T_w=1800 K, P_{in}=10 atm, and P_{out}=9.5 atm, Q=50 MW

The nozzle configuration used is shown in a schematic diagram in Figure 6.31, and the flow conditions utilized are as follows: The inlet stagnation pressure is 20 atm; the inlet gas temperature is 4000 K; the constant wall temperature is kept at 1800 K; and UF_4 gas properties evaluated at 2000 K are used. A mesh is generated that has 60 points in the streamwise direction and 60 points in the radial direction for the lower half of the nozzle due to the geometric symmetry about the center line. In the radial direction the mesh is compressed near the solid surface of the lower boundary to resolve the viscous layer. The upper boundary of the simulation is the nozzle centerline. Figure 6.32 shows the grid system mentioned above. Results of the simulation are presented in terms of contours, velocity vector plots, pressure distribution and heat transfer rates along the solid wall.

Figure 6.33 depicts the flow pattern which indicates that the flow is accelerated through the throat of the nozzle. A close view of the velocity and the temperature at various locations along the solid wall are presented in Figures 6.34 and 6.35. The hydrodynamic and thermal boundary layers are observed in both figures. Figures 6.36 and 6.37 show the Mach number contours and the pressure contours for the nozzle. The subsonic inlet flow accelerates from left to right and reaches sonic speeds near

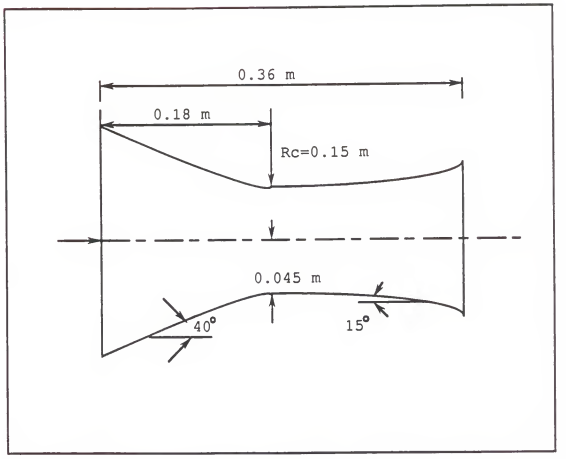


Figure 6.31 A converging-diverging nozzle

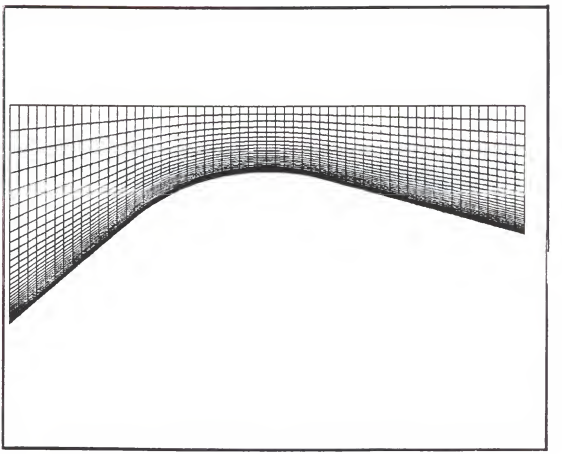


Figure 6.32 A 60x60 grid in a converging-diverging nozzle.

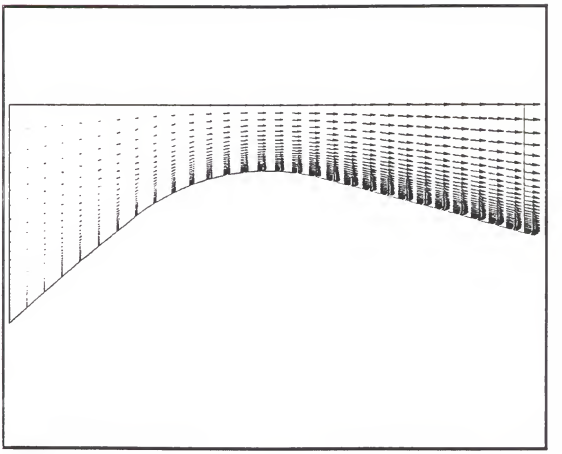


Figure 6.33 Flow pattern in a converging-diverging nozzle.

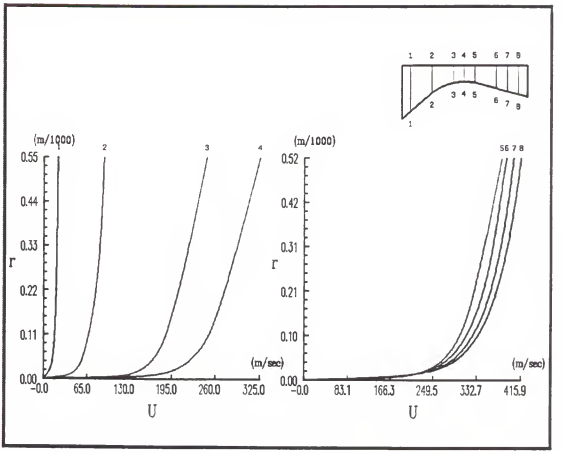


Figure 6.34 Close-up view of velocity distribution.

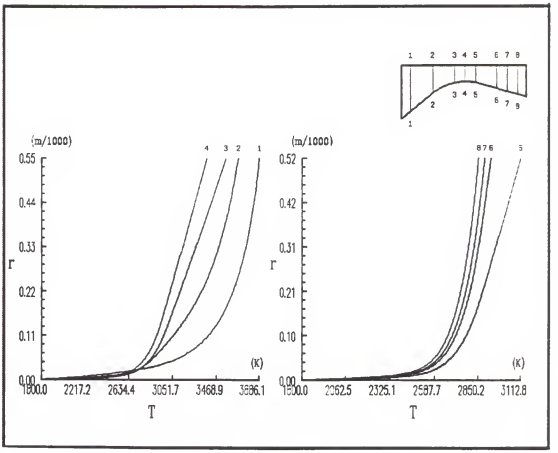
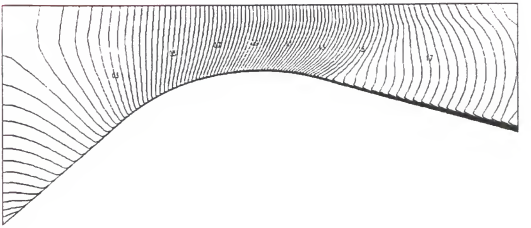
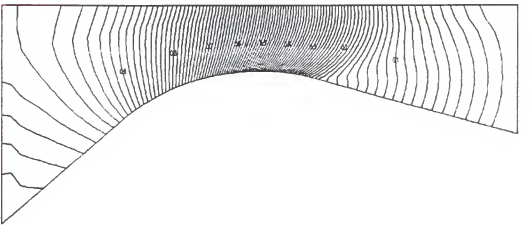


Figure 6.35 Close-up view of temperature distribution.



(a)



(b)

Figure 6.36 contours.
(a) isomach number contour.
(b) normalized pressure contour.

the throat and continues to accelerates as the flow proceeds downstream. As can be seen from both the pressure and the Mach number contour, compression waves are initiated along the wall surface just downstream from the throat toward the axis of symmetry. These waves coalesce to form a weak oblique shock which is further propagated as it approaches the axis. Moreover, the presence of an oblique shock at an axial location just downstream from the throat is clearly shown in the normalized pressure distribution in which the static-to-stagnation pressure ratio is plotted as a function of axial distance, L/D_{thrt}, along the nozzle in Figure 6.37.

6.2.2 Convective and Radiative Heat Transfer

The variation of the heat fluxes along the nozzle are shown in Figure 6.38 for the same flow conditions given in the previous section and the nozzle shown in Figure 6.31. Figure 6.38 reveals that the maximum value of the convective heat flux occurs just upstream of the throat in the vicinity of the mass flux maximum. The radiative heat flux exhibits a trend similar to that of the convective heat flux because both fluxes are proportional to the same temperature gradient at the wall. With the selection of higher stagnation pressures, 50 atm and 100 atm, for the inlet boundary condition, it is evident from Figure 6.39 that the heat fluxes increase, as expected, with increasing

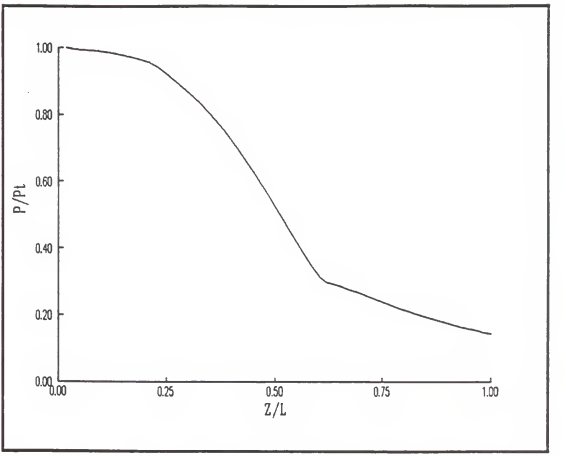


Figure 6.37 Normalized pressure distribution along the nozzle surface.

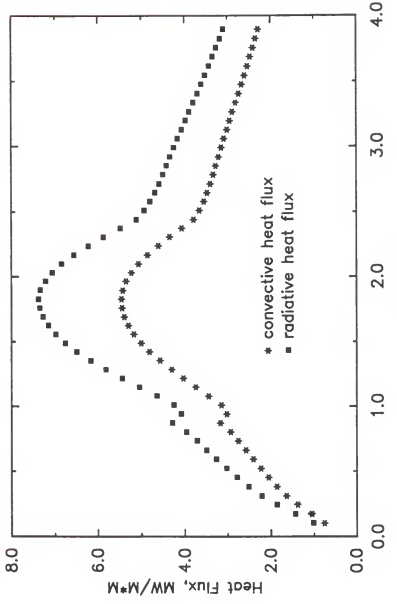


Figure 6.38 Convective and radiative heat fluxes for flow conditions. (T_{in}=4000 K, T_w=1800 K, P_{in}=20 atm, and P_{out}=1 atm)

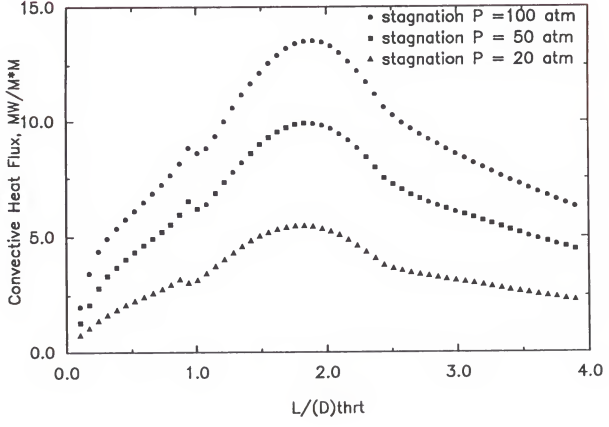


Figure 6.39 Convective heat flux for different stagnation inlet pressures.

stagnation pressures which correspond to larger mass fluxes. The variation of heat fluxes along the nozzle as a function of changes in geometrical parameters of the nozzle, such as a throat curvature and an uphill angle of the nozzle, is also examined. The obtained heat fluxes decrease with an increase of the uphill angle toward the entrance of the nozzle, as shown in Figure 6.40, but there is no significant change of the maximum heat flux at the throat. Figure 6.41 presents results for changes in the throat curvature. As shown in Figure 6.41, the heat fluxes are only broadened by increasing the throat curvature, while the peak value of the heat flux at the throat remains unchanged. Thus, it can be stated that changes in the geometrical parameters do not affect the maximum heat flux at the throat. In all the figures of the heat fluxes, small peaks in the middle of the upstream region are observed. These peaks are a result of the existence of weak secondary flows.

6.2.3 Effect of Grid System

As previously mentioned, a disadvantage of algebraic methods for grid generation is that they tend to yield skewed grid systems, which may affect the flow and thermal field solution and, furthermore, may create a more severe problem, such as a convergence problem in a time-marching scheme. Thus, the effect of the grid system for the solution at the steady state is investigated by comparing

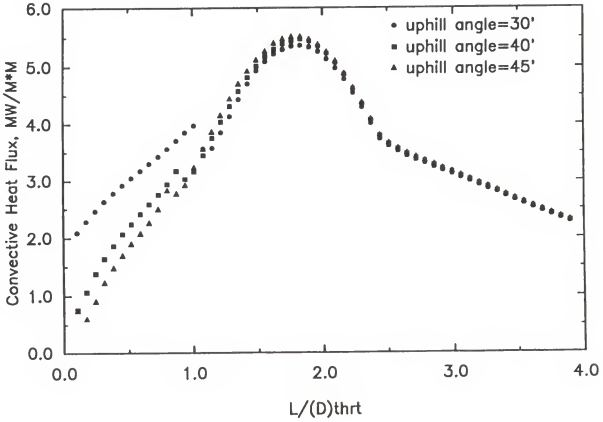


Figure 6.40 Convective heat flux for different uphill angles of the upstream region.

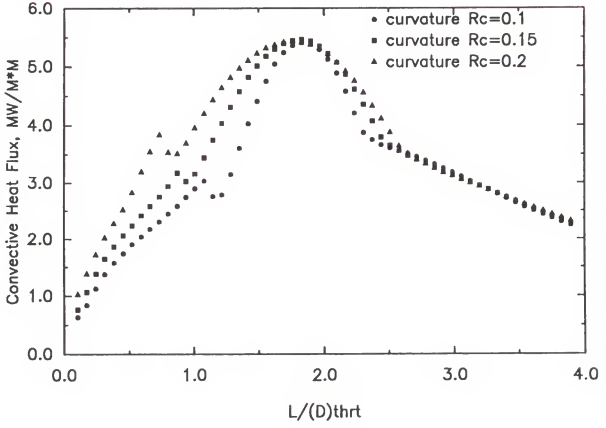


Figure 6.41 Convective heat flux for different radii of curvature at throat.

results obtained from both the skewed and orthogonal grid systems. The skewed and orthogonal grid system for a bell-type nozzle are displayed in Figures 6.42 and 6.43. Flow conditions are the same as those of the previous section, but a different choice of grid size, 40x60, is used. Figures 6.44 (a) and (b) represent the flow patterns for the skewed and the orthogonal grid system, respectively. An additional comparison is performed with the contours of pressure, density, velocity, and Mach number. The contours for both grid systems are presented in Figures 6.45 through 6.48. From these figures, it is observed that the results obtained from the different grid systems are almost the same. Further, the static-to-stagnation pressure ratios along the nozzle in Figures 6.49 (a) and (b) are equal. In conclusion, there is no advantage in using an orthogonal grid system over using a skewed grid system for the simple nozzle configuration examined in this work.

6.2.4 Assessment for Using Frequency Averaged Rosseland Absorption Coefficient in Radiative Heat Transfer

In this study, frequency averaged absorption coefficients are used to predict radiative transfer. Using the mean absorption coefficient eliminates carrying out a spectral analysis and integrating over all wavelengths to obtain total energy quantities. The question is whether the analysis will yield an accurate solution for a particular problem. Thus, an analysis is performed to assess the

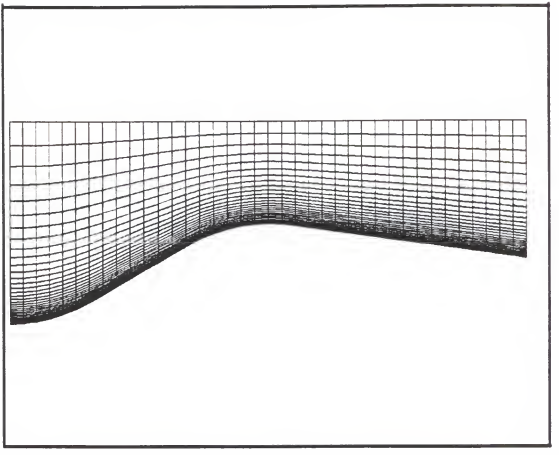


Figure 6.42 A skewed grid in a bell-type nozzle.

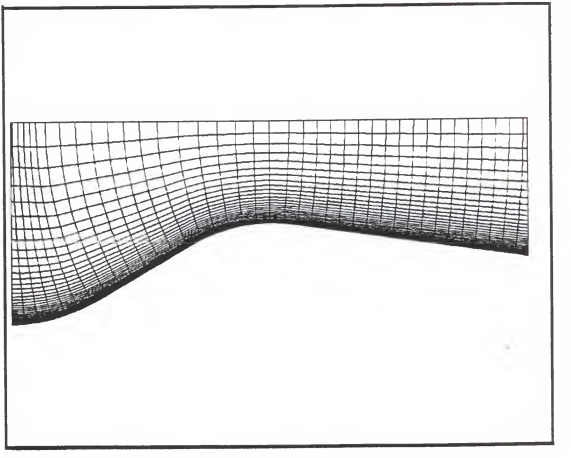
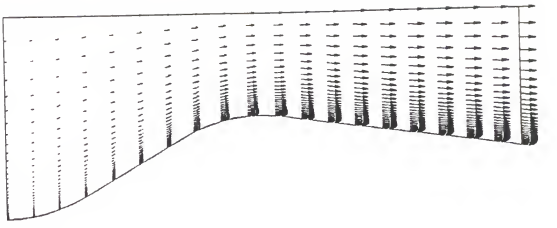
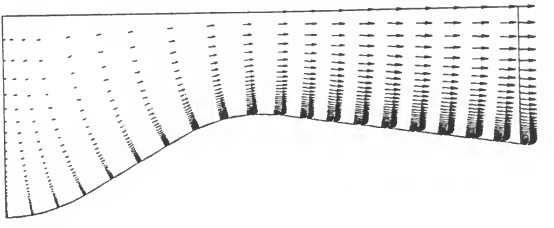


Figure 6.43 An orthogonal grid in a bell-type nozzle.

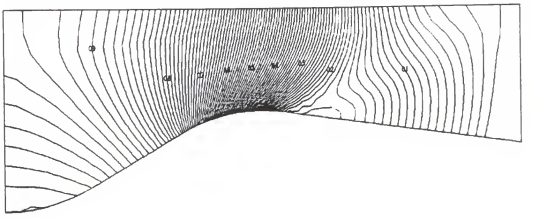


(a)

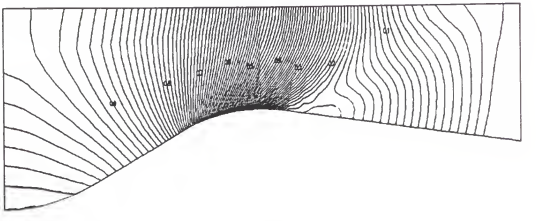


(b)

Figure 6.44 Flow patterns.
(a) skewed grid
(b) orthogonal grid

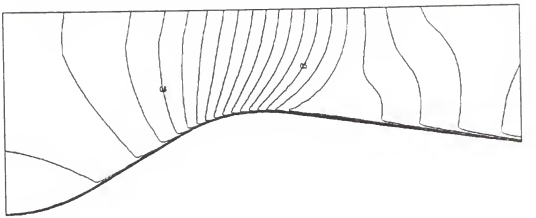


(a)

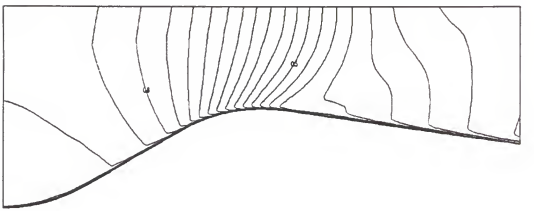


(b)

Figure 6.45 Normalized pressure contours.
(a) skewed grid
(b) orthogonal grid

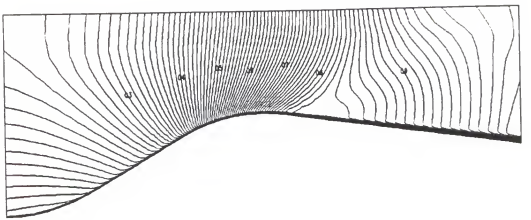


(a)

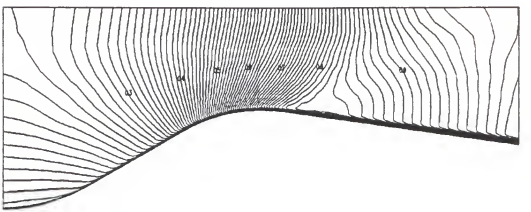


(b)

Figure 6.46 Normalized density contours.
(a) skewed grid
(b) orthogonal grid

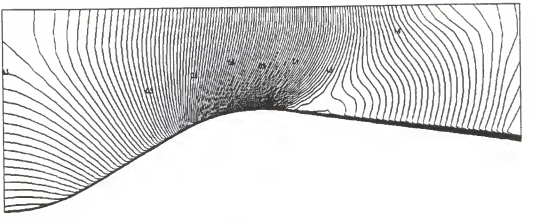


(a)

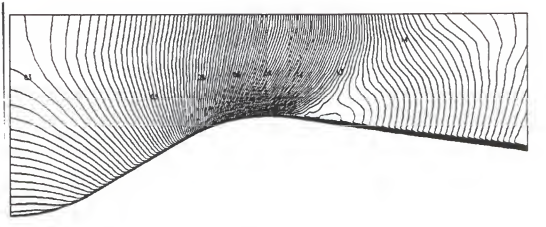


(b)

Figure 6.47 Normalized velocity contours.
(a) skewed grid
(b) orthogonal grid

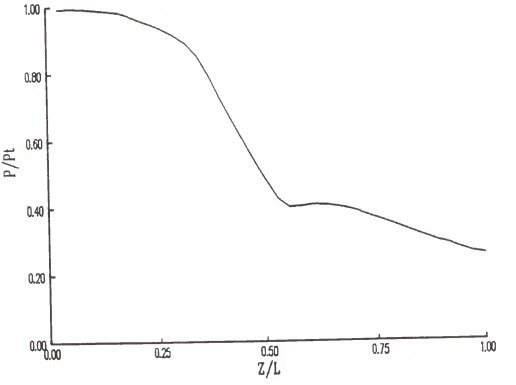


(a)

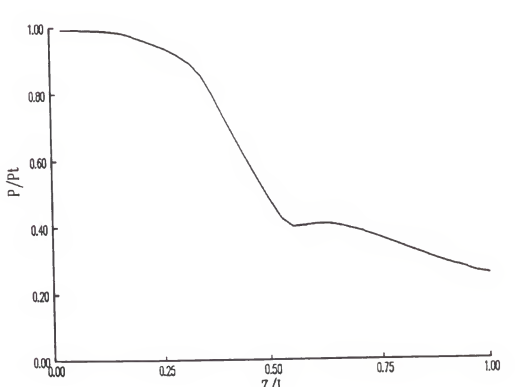


(b)

Figure 6.48 Mach number contours.
(a) skewed grid
(b) orthogonal grid



(a)



(b)

Figure 6.49 Surface normalized pressure distribution along the nozzle.
(a) skewed grid
(b) orthogonal grid

sensitivity of radiative transfer to the mean absorption coefficient. The absorption spectrum of UF_6 from 2000 to 4200 Å [31] is used. The analysis is performed with a mean absorption coefficient over all the spectral range and mean absorption coefficients for two wavelength bands, 2000 Å-3000 Å and 3000 Å-4200 Å. The Rosseland mean opacity over all wavelengths, 2000 Å-4200 Å, is 5.0×10^3 (1/m) for temperature, 4000 K, and pressure, 10 atm. The Rosseland mean opacities for the wavelength bands of 2000 Å-3000 Å and 3000 Å-4200 Å are 1.06×10^4 (1/m) and 3.8×10 (1/m), respectively.

Using the above opacities, radiative heat fluxes are estimated with the diffusion approximation and 1-D integral approximation. As expected, the diffusion approximation is very sensitive to the magnitude of the opacity. For the wavelength range of 2000 Å-3000 Å, the radiative heat flux is underpredicted as plotted in Figure 6.50. For the range of 3000 Å-4200 Å, the diffusion approximation is not applicable because the criterion for the diffusion approximation, described in subsection 2.2.1, is not satisfied by the long mean free path of photon. As seen in this analysis, the Rosseland mean absorption coefficient over all wavelengths may have a large value, but the spectral absorption coefficient may be very small in certain spectral regions. Therefore, it is recommended to use wavelength bands in which the spectral absorption

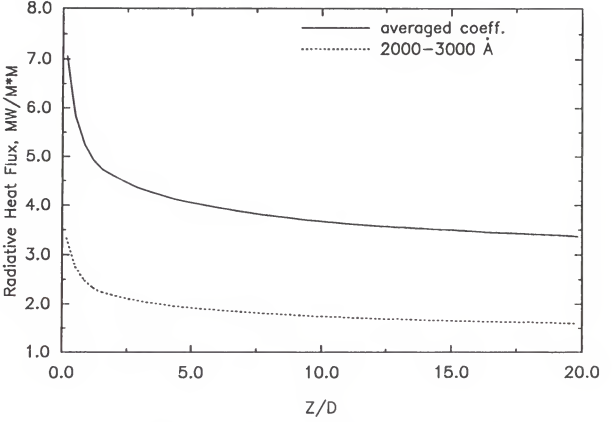


Figure 6.50 Radiative heat transfer rates calculated by the diffusion approximation for a mean opacity over all wavelengths and for a mean opacity over the range of 2000 Å–3000 Å.

coefficient is everywhere large and to evaluate a Rosseland mean for each of these regions.

The 1-D integral approximation is less sensitive to the magnitude of the Rosseland mean opacity. Radiative heat transfer rates are calculated with the mean opacity over all wavelengths and the mean opacities for the ranges of 2000 Å-3000 Å and 3000 Å-4200 Å. The result for the mean opacity is compared with the averaged heat flux for the opacities of two wavelength bands. Figure 6.51 shows the results and reveals that use of the mean opacity for 1-D integral approximation can be acceptable for the prediction of radiative heat transfer.

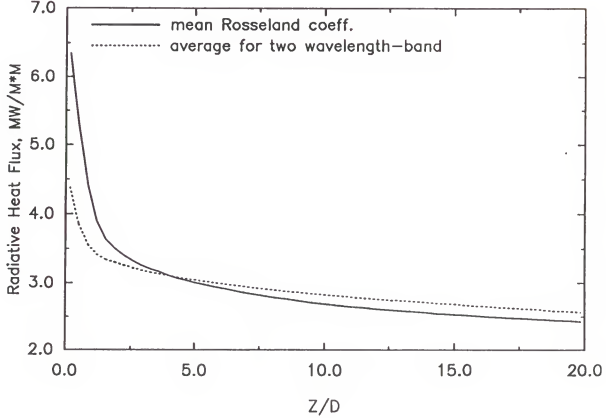


Figure 6.51 Radiative heat transfer rates calculated by the 1-D integral approximation for a mean opacity over all wavelengths and an averaged result for mean opacities over the ranges of 2000 Å-3000 Å and 3000 Å-4200 Å.

CHAPTER 7
SUMMARY AND CONCLUSIONS

An investigation into the use of computational fluid dynamics (CFD) has been carried out to predict the flow field solutions and expected heat transfer rates of interest to Ultrahigh Temperature Vapor Core Reactor systems using a straight tube geometry and the converging-diverging geometry of the reactor exit nozzle. An axisymmetric two-dimensional (2-D) Navier-Stokes based model, which is composed of an algebraic two-layer eddy viscosity turbulence model, a convective heat transfer model, and radiative heat transfer models, is developed. Using the developed models, the following studies have been accomplished.

1. The flow and thermal fields at a high Reynolds number and at a high temperature are expressed in a thin-layer Navier-Stokes equations. The governing equations are solved by a hybrid implicit-explicit method based on the finite volume approach. The numerical algorithm used in this study has a rapid convergence compared to a simple explicit method. Even though very fine grids are used for the prediction of convective heat flux into the wall, less than 300 iterations are required to reach the steady-state for any flow conditions taken. The computed velocity

distribution in a straight tube for air is compared to existing experimental data. Good agreement between the computed results and the measured data indicates the validity of the developed model and the related assumptions for prediction of flows.

2. Applicability for transonic flow modeling is examined by checking the capability of capturing shock waves in a channel with an inward circular arc bump. The computed data of the Mach number and pressure are presented in Mach number contours and normalized pressure contours, which show clear, sharp shock waves located around the trailing edge of the bump. Applicability for transonic flow modeling is also assured by observing the discontinuity due to a shock wave in the surface pressure distribution.

3. Convective heat flux into the wall is directly determined by the known temperature gradients at the wall if extremely fine grids contain a grid point within the laminar sublayer, $y^+ \approx 5$. The computed heat transfer rates in the Nusselt number are compared to the measured data for air and the result obtained with an empirical correlation, Dittus-Boelter correlation. Good agreements between the computed result and the measured data and the result of the empirical correlation indicate that the developed flow and heat transfer model is accurate.

4. Radiative heat flux is predicted by both the diffusion approximation and the one-dimensional equation of

radiative transfer. From the comparison of the two results obtained, it is found that the radiative heat flux obtained by the diffusion approximation is underestimated by about 30% compared to the other results for mean absorptivities greater than 555.5 cm^{-1}, evaluated at the stagnation pressure, 50 atm, and the wall temperature, 1800 K.

5. The effect of internal heat generation on heat transfer in straight tubes is examined for a variety of power densities, 0 to 1000 w/cc. For very high power densities such as 500 w/cc and 1000 w/cc, the heat supplied by the internal heat sources considerably exceeds the heat transferred through the wall.

6. The convective and radiative heat fluxes are predicted for a converging-diverging nozzle. The maximum value of convective and radiative heat fluxes, as expected, occurs just upstream of the throat in the vicinity where the mass flux is a maximum. The heat fluxes, in general, increase with increasing stagnation pressures which correspond to larger mass fluxes.

The above accomplishments demonstrate that the computational fluid dynamics and heat transfer model based on the axisymmetric, thin-layer, Navier-Stokes equations represents a viable computational tool for predicting flow and thermal fields of interest pertaining to UTVR systems. Moreover, the model developed can be applied to many engineering applications related to energy system, such as

combustion chambers, Gas Core Reactors (GCR), and nuclear propulsion systems.

Convective heat flux is accurately predicted by the convective heat transfer model, but an assessment of radiative heat transfer models in comparison with experimental data is not made because experimental data of radiative heat transfer rates for an optically thick gas are not available. Therefore, it is desirable that hardware be built to test the radiative heat transfer rates obtained by the radiative heat transfer models.

APPENDIX
SOURCE LISTING OF CODE

This is the source program of a Navier-Stokes based model for the analysis of the combined radiative and convective heat transfer.

```
c-----------------------------------------------------------------------
      program main
c-----------------------------------------------------------------------
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      common/b/iturb,prt,ep(100,100)
      common/c/du(100,100,4),dus(100,100,4)
      common/e/g11(101),g12(101),g13(101),g14(101),
     1         g21(101),g22(101),g23(101),g24(101),
     2         g31(101),g32(101),g33(101),g34(101),
     3         g41(101),g42(101),g43(101),g44(101),
     4         ff(101),implt
       common/splt/iflxs,sz,sr
       common/q/qp
       common/gm/thrt
       dimension ynu(100),znum(100),rr(100)
       common/rad/nradq,rabsco
       dimension pu(100,100),pv(100,100),newtw(100)
       dimension rnr(100),u17(100),u110(100)
       common/tem/t(101,101),ttwall
       common/speed/rmach(100,100)

c-----------------------------------------------------------------------
c     gamma - ratio of specific heats, cp/cv
c     cv - specific heat at constant volume
c     pr - prandtl number
c     prt - turbulent prandtl number
c     iflxs = 0 no flux splitting
```

```
c          = 1 first order flux splitting included
c          = 2 second order flux splitting included
c     il - range of mesh index i
c     jl - range of mesh index j
c     nend - range of time step index nadv
c     nprnt - print parameter (complete flow field printed at
c     nadv=nend,surface properties printed at nadv=k*nprnt,
c     for k=1,2,3...,nprnt must be a divisor of nend)
c     cfl - courant-friedrich-lewy number (must be less than one)
c     iread = 0 - new start
c           = 1 - start and continue from stored file
c     iwrite = 0 - no data file to be written for restarting
c            = 1 - data file to be written for restarting
c     implt = 0 - explicit maccormack method
c           > 0 - implicit gauss seidel line relaxation
c     iturb = 0 - laminar flow, = 1 - turbulent flow
c    nvisc - value of nadv at which viscous flow computation begins
c     jlfm - estimate of range of j index containing boundary layer
c     iadbwl = 1 - adiabatic wall, = 0 - isothermal wall
c     twall - temperature of wall
c     ptt - total pressure specified at entrance
c     ttt - total temperature specified at entrance
c     vou - tangent of flow angle at entrance
c     pe  - pressure specified at exit
c     qp - internal heat source(w/m**3)
c     nradq = 0 - no radiation heat transfer
c           = 1 - radiation heat transfer
c     rabsco - rosseland absorption coefficient
c     igrid = 0 - skew grid
c           = 1 - orthogonal grid
c-----------------------------------------------------------------------
c
        open(unit=15,file='tube.inp',status='old')
        open(unit=16,file='tube.out',status='old')
        open(unit=22,file='univ.dat',status='old')
        open(unit=50,file='graph.dat',status='old')
        open(unit=999,file='sub1.dat',status='old')
        open(unit=77,file='heat2.dat',status='old')
        open(unit=60,file='resdu.dat',status='old')
c
c     gamma is 1.4 for air and gamma is 1.08 for UF4.
c       gamma=1.4
       gamma=1.08
       gm1=gamma-1.0
       ggm1=gamma*gm1

c     cv is 319.31 for UF4 property estimated at 2000 k in j/kg k.
c     cv is 717.5 for air in j/kg K.
      cv=319.31
c       cv=717.5
c     Prandtl number of uf4 is 0.915.
c     Prandtl number of air is 0.72
```

```
      pr=0.915
c      pr=0.72
      prt=0.90
      read (15,*) il,jl,nend,nprnt,iread,iwrite,iflxs,igrid
      read (15,*) implt,iturb,nvisc,jlfm,nradq,rabsco,ttwall
      read (15,*) iadbwl,twall,cfl,vou,ptt,ttt,pe,qp
c
      il=il+mod(il,2)
      ilm1=il-1
      jlm1=jl-1
      eiwall=cv*twall
      do 1 i=1,il
      do 1 j=1,jl
      ep(i,j)=0.0
    1 continue
c
c     Initial imformations are stored for graphics.
c
      write(50,*) nend
      write(50,*) nprnt
      write(50,*)il
      write(50,*)jl
      write(50,*)m
      write(50,*)ptt
c
      if (iread.eq.0) call gmtry
c
      call prntxy
c
      time=0.0
      nadv=0
      if (iread.eq.0) call initl
      if (iread.eq.1) call rdwrt(1,time)
      nstart=nadv+1
      lcnt=1
      write (16,5)
    5 format(1h1)
      write (16,10)
   10 format(///,56x,18hi n p u t  d a t a,//)
      write (16,20) il,iturb,cfl
   20 format(1h ,30x,8hil     =,2x,i8,7x,8hiturb  =,2x,i8,7x,
     c  8hcfl    =,2x,f8.3)
      write (16,30) jl,iadbwl,iflxs
   30 format(1h ,30x,8hjl     =,2x,i8,7x,8hiadbwl =,2x,i8,7x,
     c  8hiflxs  =,2x,i8)
      write (16,40) nend,twall,nstart
   40 format(1h ,30x,8hnend   =,2x,i8,7x,8htwall  =,2x,f8.3,7x,
     c  8hnstart =,2x,i8)
      write (16,50) nprnt,nvisc,implt
   50 format(1h ,30x,8hnprnt  =,2x,i8,7x,8hnvisc  =,2x,i8,7x,
     c  8h implt =,2x,i8)
      write (16,80)
```

```
   80 format(////,51x,28hupstream boundary conditions)
      write (16,90)
   90 format(/,32x,19x,3hptt,19x,3httt,19x,3hvou)
      write (16,100) ptt,ttt,vou
  100 format(32x,3(10x,f12.4))
      m=1
      n=2
c
      call bc
      if (nend.eq.0.and.iwrite.eq.1) call rdwrt(2,time)
      if (nend.eq.0) stop
      k1=1
      k2=2
c
      do 111 i = 1,il
        do 111 j = 1,jl
        pu(i,j)=0.0
        pv(i,j)=0.0
111     continue
c
      do 200 nadv=nstart,nend
      do 140 i=1,il
      do 140 j=1,jl
      si=sqrt((z(i,j+1)-z(i,j))**2+(r(i,j+1)-r(i,j))**2)
      szp=+(r(i,j+1)-r(i,j))/si
      srp=-(z(i,j+1)-z(i,j))/si
      uu=sxp*u(i,j)+syp*v(i,j)
      sj=sqrt((z(i+1,j)-z(i,j))**2+(r(i+1,j)-r(i,j))**2)
      sxp=-(r(i+1,j)-r(i,j))/sj
      syp=+(z(i+1,j)-z(i,j))/sj
      vv=szp*u(i,j)+srp*v(i,j)
      cc=sqrt(ggm1*abs(ei(i,j)))
      dti=0.5*vol(i,j)/si/(abs(uu)+cc)
      dtj=0.5*vol(i,j)/sj/(abs(vv)+cc)
      if (i.eq.1.and.j.eq.1) then
      dte=dti
      dt=cfl*dti
      endif
      if (dti.lt.dte) dte=dti
      if (dtj.lt.dte) dte=dtj
  140 continue
      if (implt.eq.0)  dt=cfl*dte
      if(mod(nadv,nprnt).eq.0) write(16,150) nadv,dte,dt
  150 format(10x,5hnadv=,i7,5x,4hdte=,e12.4,5x,3hdt=,e12.4)
      if (nadv.ge.nvisc.and.iturb.eq.1) call turbfl
      time=time+dt
      call l(dt)
      sum=0.0
        do 300 i =1,il
        do 300 j =1,jl
      sum=sum+sqrt((u(i,j)-pu(i,j))**2 + (v(i,j)-pv(i,j))**2)
300     continue
```

```
        sum=sum/(il*jl)
        do 400 i =1,il
        do 400 j =1,jl
        pu(i,j)=u(i,j)
        pv(i,j)=v(i,j)
400     continue
c        if(mod(nadv,nprnt).eq.0.or.nadv.eq.nend)then
         do 155 i=1,il
         do 155 j=1,jl
         t(i,j)=ei(i,j)/cv
155      continue
c
c
c -  calculate mach number
         do 157 i=1,il
         do 157 j=1,jl
         c=sqrt(ggm1*abs(ei(i,j)))
         rmach(i,j)=sqrt(u(i,j)*u(i,j)+v(i,j)*v(i,j))/c
157      continue
       if (mod(nadv,nprnt).eq.0) write(16,160) nadv,dt,time
         write(6,*) 'iter #=',nadv,'  ,   residual=',sum
         write(60,*) nadv,sum

  160 format(10x,5hnadv=,i7,5x,3hdt=,e12.4,5x,5htime=,e12.4)
      if (mod(nadv,nprnt).eq.0.or.nadv.eq.nend) call prntff(time)
  200 continue

c    ------------------------------------------------------------
c        calculation of heat flux
c    ------------------------------------------------------------
c    ------------------------------------------------------
c      first order error  heat flux calculation
c    ------------------------------------------------------
        dd=thrt*2.
        ilm3=il-3
        do 153 i=2,il-1
        dr=0.5*(r(i,3)-r(i,2))
        dt=t(i,2)-twall
        dtt=t(i,2)-twall
        dtdr=dt/dr
        dttdr=dtt/dr
        dr1=(r(i,4)-r(i,2))+0.5*(r(i,5)-r(i,4))
        dr2=(r(i,8)-r(i,2))+0.5*(r(i,9)-r(i,8))
        dr3=(r(i,12)-r(i,2))+0.5*(r(i,13)-r(i,12))
        d1=dr
        d2=(r(i,3)-r(i,2))+0.5*(r(i,4)-r(i,3))
        d3=dr1
        d4=(r(i,5)-r(i,2))+0.5*(r(i,6)-r(i,5))
        d5=(r(i,6)-r(i,2))+0.5*(r(i,7)-r(i,6))
        d6=(r(i,7)-r(i,2))+0.5*(r(i,8)-r(i,7))
        d7=dr2
        d8=(r(i,9)-r(i,2))+0.5*(r(i,10)-r(i,9))
```

```
        d9=(r(i,10)-r(i,2))+0.5*(r(i,11)-r(i,10))
        d10=(r(i,11)-r(i,2))+0.5*(r(i,12)-r(i,11))
        d11=dr3
        d12=(r(i,13)-r(i,2))+0.5*(r(i,14)-r(i,13))
        d20=(r(i,20)-r(i,2))+0.5*(r(i,21)-r(i,20))
        dt1=t(i,4)-twall
        dt2=t(i,8)-twall
        dt3=t(i,12)-twall
        dt20=t(i,20)-twall
        dtdr1=dt1/dr1
        dtdr2=dt2/dr2
        dtdr3=dt3/dr3
        dtdr20=dt20/d20
        do 1110 j=1,jl-1
        if(j.eq.1) then
        rr(j)=0.5*(r(i,3)-r(i,2))
     else
        rr(j)=(r(i,j+1)-r(i,2))+0.5*(r(i,j+2)-r(i,j+1))
     endif
1110 continue
        if(i.eq.ilm3)then
        write(6,*)'1,dr',i,dr,dr1,dr2,dr3
        write(6,*)'d1 to d6',d1,d2,d3,d4,d5,d6
        write(6,*)'d7 to d12',d7,d8,d9,d10,d11,d12
  endif

c     - air viscosity (Sutherland's formula)
c     rmu=1.458e-06 * sqrt(t(i,2)**3)/(t(i,2)+110.4)
c     - uf4 viscosity at 2000 K
c     rmu=3.357e-6*sqrt(t(i,2))/(aa+bb*t(i,2))
      rmu=8.67e-5

c     - thermal conductivity rk=cv*gamma*rmu/pr
        rk=cv*gamma*rmu/pr
        zod=0.5*(z(i,2)+z(i+1,2))/dd

c     ------------------------------------------------------
c     second order error heat flux calculation
c     ------------------------------------------------------
c     dd=2.*thrt
c     do 153 i=2,il-2
c     drmins=0.5*(y(i,3)-y(i,2))
c     dr=0.5*(r(i,4)-r(i,3))
c     drplus=dr+drmins
c     alpa=drplus/drmins
c  - calculate radiative heat flux
      if(nradq.eq.1)then
      sigma=5.67e-8
c     rabsco-rosseland absorption coefficient-
c     tt=0.5*(t(i,2)+t(i,3))
      radk=(8.*sigma*(ttwall**3))/(3.*rabsco*2.)
      endif
```

```
c     dtdy=(-alpa*(alpa+2.)*twall+((alpa+1.)**2)*t(i,2)-t(i,3))/
c      1        (alpa*(alpa+1.)*dymins)
c
c     - qd is the variable of convective heat transfer rate at the
c       wall.
c     - qrad is the variable of radiative heat transfer rates at the
c       wall
c
      qd=1.e-6*rk*dtdy
      if(nradq.eq.1)then
         qrad=1.e-6*radk*dtdy
         qtot=qd+qrad
         zod=0.5*(x(i,2)+x(i+1,2))/dd
         write(77,*) zod,qd,qrad,qtot
         endif
153      continue

c     - if 1-D integaral approxiamation is used to calculate the
c       radiative heat transfer rate, this parts must be included.
c       aaw=rabsco*(ttt/ttwall)
c       do 2222 i =2, il-1
c       plusin=0.0
c       do 2223 j =2, jl-2
c       rw=r(i,j)-r(i,2)
c       if(rw.le.(5./aaw))then
c       rr(j)=0.5*(r(i,j+2)-r(i,j)
c       ri=((t(i,j)+t(i,j+1))*0.5)**4
c       plusw(i,j)=rabsco*sigma*(1./pi)*ri*exp(-rabsco*rr(j))*rr(j)
c       plusin=plusin+plusw(i,j)
c       endif
c2223   continue
c       zod=0.5*(z(i,2)+z(i+1,2))/dd
c       plusi(i)=plusin
c       radw(i)=1.e-6*(plus(i)-(1./pi)*sigma*(twall)**4))
c       when 1-D integral approximation is used, make an
c       open statement
c       write(1000,*) zod, radw(i)
c2222   continue
c
      if (iwrite.eq.1) call rdwrt(2,time)
      stop
      end

c-----------------------------------------------------------------------
      subroutine gmtry
c-----------------------------------------------------------------------
c
c     reads computational mesh coordinates
c
     common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c        e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
```

```
      c          fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
      c          gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
      c          il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
      c          iadbwl,eiwall,lcnt,cfl,
      c          nadv,nend,nvisc
      common/gm/thrt
      common/grid/igrid
      dimension rr(100), o(100)
c
       pi=3.14159
c     - mks unit is used.
c
       rl=1.0
       thrt=0.025
c
c     - For a converging-diverging nozzle, rad is a radius of c
c       curvature at throat.
c
c       rad=0.02
c     - theta is uphill angle and epsln is downhill angle.
c
c       theta=20.0*pi/180.0
c
c     - For a straight tube, both angles are zero.

        theta=0.0

c     - epsln=20.0*pi/180
        epsln= 0.0
c
       ithrt=il/2+1
       sint=sin(theta)
       cost=cos(theta)
       sine=sin(epsln)
       cose=cos(epsln)
       dr=thrt/1.0e+4
c
       deta=1.0/(jl-2)
       call stretch(thrt,dy,deta,ck,ekm1)
c
c     cd=ck*deta
       do 1 j=2,jl
c
       r(ithrt,j)=-thrt*(1.0-(exp((j-2)*ck*deta)-1.0)/ekm1)
c
     1 continue
c
c
       dz=1.0*rl/(il-2)
       detap=1.0/(ithrt-2)
       call stretch(0.5*rl,dz,detap,ckp,ekm1p)
       do 2 i=2,il
```

```
      if (i.le.ithrt) then
      z(i,2)=0.5*rl*(1.0-(exp((ithrt-i)*ckp*detap)-1.0)/ekm1p)
      else
      z(i,2)=0.5*rl*(1.0+(exp((i-ithrt)*ckp*detap)-1.0)/ekm1p)
      endif
      if (z(i,2).le.0.5*rl-rad*sint)
     c    yri,2)=-thrt-rad+rad*cost
     c          -(0.5*rl-rad*sint-z(i,2))*sint/cost
      if (z(i,2).gt.0.5*rl-rad*sint.and.z(i,2).lt.0.5*rl+rad*sine)
     c    r(i,2)=(-(thrt+rad)+sqrt(rad*rad-(z(i,2)-0.5*rl)**2))
      if (z(i,2).ge.0.5*rl+rad*sine)
     c    r(i,2)=-thrt-rad+rad*cose
     c          -(z(i,2)-0.5*rl-rad*sine)*sine/cose
      do 2 j=3,jl
      z(i,j)=z(i,2)
      r(i,j)=-r(i,2)*r(ithrt,j)/thrt
    2 continue
c
c     set boundary cells by linear extrapolation
c
      do 10 i=2,il
        z(i,1)=2.0*z(i,2)-z(i,3)
        r(i,1)=2.0*r(i,2)-r(i,3)
        z(i,jl+1)=2.0*z(i,jl)-z(i,jl-1)
        r(i,jl+1)=2.0*r(i,jl)-r(i,jl-1)
   10 continue
c
c
      jlp1=jl+1
      do 20 j=1,jlp1
        z(1,j)=2.0*z(2,j)-z(3,j)
        r(1,j)=2.0*r(2,j)-r(3,j)
        z(il+1,j)=2.0*z(il,j)-z(il-1,j)
        r(il+1,j)=2.0*r(il,j)-r(il-1,j)
   20 continue
c
c    calculate orthogonal grid system by an algebraic technique.
      if(igrid.eq.0) then
      go to 41
      else
      do 300 i =3, ithrt-1
      teta=atan((r(i+1,2)-r(i,2))/(z(i+1,2)-z(i,2)))
      rr(i)=(r(i,jl)-r(i,2))/sin(teta)
      o(i)=z(i,jl)+(r(i,jl)-r(i,2))/tan(teta)
      do 300 j =3,jl
      phi=atan((r(i+1,j)-r(i,j))/(z(i+1,j)-z(i,j)))
      z(i,j)=o(i)-rr(i)*cos(phi)
      r(i,j)=-(o(i)-z(i,j))*tan(phi)
300   continue
      do 301 i=ithrt+1, il-1
      teta=atan((r(i+1,2)-r(i,2))/(z(i+1,2)-z(i,2)))
      rr(i)=(r(i,jl)-r(i,2))/sin(teta)
```

```
        o(i)=z(i,jl)+(r(i,jl)-r(i,2))/tan(teta)
        do 301 j =3,jl
        phi=atan((r(i+1,j)-r(i,j))/(z(i+1,j)-z(i,j)))
        z(i,j)=o(i)-rr(i)*cos(phi)
        r(i,j)=-(o(i)-z(i,j))*tan(phi)
301     continue
        endif
41      continue
c
c       calculate all volumes
c
        do 30 j=1,jl
        do 30 i=1,il
            vol(i,j)=(abs((z(i,j)-z(i+1,j))*r(i+1,j+1)
     1                  +(z(i+1,j)-z(i+1,j+1))*r(i,j)
     2                  +(z(i+1,j+1)-z(i,j))*r(i+1,j))
     3              +abs((z(i,j)-z(i+1,j+1))*r(i,j+1)
     4                  +(z(i+1,j+1)-z(i,j+1))*r(i,j)
     5                  +(z(i,j+1)-z(i,j))*r(i+1,j+1)))/2.0
   30 continue
c
c       reset the volumes of the boundary cells equal to that of
c       their flowfield counterparts.
c
c
        do 40 i=1,il
        vol(i,1)=vol(i,2)
        vol(i,jl)=vol(i,jlm1)
   40 continue
        return
        end
c
c-------------------------------------------------------------------
        subroutine stretch(rr,dd,deta,ck,ekm1)
c-------------------------------------------------------------------

        ck=1.0
        do 10 k=1,10
c        do 10 k=1,18
        ek=exp(ck)
        ekdeta=exp(ck*deta)
        fk=dd-rr*(ekdeta-1.0)/(ek-1.0)
        fkp=-rr*(deta*ekdeta-(ekdeta-1.0)*ek/(ek-1.0))/(ek-1.0)
        ck=ck-fk/fkp
   10 continue
        ekm1=exp(ck)-1.0
        return
        end
```

```
c-----------------------------------------------------------------------
      subroutine initl
c-----------------------------------------------------------------------
c
c     form initial flow field
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c        e(100,100,2),fs1(2),fs2(2),
     c        u(100,100),v(100,100),ei(100,100),p(100,100),
     c        fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      common/e/g11(101),g12(101),g13(101),g14(101),
     c        g21(101),g22(101),g23(101),g24(101),
     c        g31(101),g32(101),g33(101),g34(101),
     c        g41(101),g42(101),g43(101),g44(101),
     c        ff(101),implt
c
      eit=cv*ttt
      rtt=ptt/(gm1*eit)
      do 10 j=1,jl
      do 10 i=1,il
      rho(i,j,1)=rtt
      rhou(i,j,1)=0.0
      rhov(i,j,1)=0.0
      e(i,j,1)=rtt*eit
      if (i.eq.il) then
      rho(i,j,1)=rtt*pe/ptt
      e(i,j,1)=pe/gm1
      endif
      u(i,j)=rhou(i,j,1)/rho(i,j,1)
      v(i,j)=rhov(i,j,1)/rho(i,j,1)
      ei(i,j)=e(i,j,1)/rho(i,j,1)-0.5*(u(i,j)**2+v(i,j)**2)
      p(i,j)=gm1*rho(i,j,1)*ei(i,j)
      rho(i,j,2)=rho(i,j,1)
      rhou(i,j,2)=rhou(i,j,1)
      rhov(i,j,2)=rhov(i,j,1)
      e(i,j,2)=e(i,j,1)
   10 continue
c
      return
      end

c-----------------------------------------------------------------------
      subroutine lbc(dt)
c-----------------------------------------------------------------------
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c        e(100,100,2),fs1(2),fs2(2),
     c        u(100,100),v(100,100),ei(100,100),p(100,100),
     c        fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
```

```
      c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
      c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
      c         iadbwl,eiwall,lcnt,cfl,
      c         nadv,nend,nvisc
       common/c/du(100,100,4),dus(100,100,4)
       common/e/g11(101),g12(101),g13(101),g14(101),
      c          g21(101),g22(101),g23(101),g24(101),
      c          g31(101),g32(101),g33(101),g34(101),
      c          g41(101),g42(101),g43(101),g44(101),
      c          ff(101),implt
       gp1=gamma+1.0
c
c     inlet bc -constant total pressure and temperature
c
       do 10 j=2,jlm1
       rtt=ptt/(gm1*cv*ttt)
       astr2=(2.0/gp1)*ggm1*cv*ttt
       rr=rho(1,j,n)
       cc=sqrt(ggm1*ei(1,j))
       uu=u(1,j)
       dd4=dt*(uu-cc)/(0.5*(z(3,j)-z(1,j)))
       r4=-(dd4/(1.0-dd4))*(p(2,j)-p(1,j)-rr*cc*(u(2,j)-u(1,j)))
       hh=ptt*(2.0*gamma/gp1)*(uu/astr2)*
      c          (1.0-(gm1/gp1)*uu*uu/astr2)**(1.0/gm1)
       um=uu-r4/(hh+rr*cc)
       rm=rtt*(1.0-(gm1/gp1)*um*um/astr2)**(1.0/gm1)
       eim=cv*ttt*(1.0-(gm1/gp1)*um*um/astr2)
       du(1,j,1)=rm-rho(1,j,n)
       du(1,j,2)=rm*um-rhou(1,j,n)
       du(1,j,3)=0.0
       du(1,j,4)=rm*(eim+0.5*um*um)-e(1,j,n)
    10 continue
c
c     exit bc subsonic-pe specified or supersonic
c
       do 20 j=2,jlm1
       si=sqrt((z(il,j+1)-z(il,j))**2+(r(il,j+1)-r(il,j))**2)
       szp= (r(il,j+1)-r(il,j))/si
       srp=-(z(il,j+1)-z(il,j))/si
       rr=rho(ilm1,j,n)
       up= szp*u(ilm1,j)+srp*v(ilm1,j)
       vp=-srp*u(ilm1,j)+szp*v(ilm1,j)
       cc=sqrt(ggm1*ei(ilm1,j))
       rccsq=1.0/(cc*cc)
       dd=si*dt/vol(ilm1,j)
       dd1=up*dd
       dd2=(up+cc)*dd
       dd4=(up-cc)*dd
       r1=-(dd1/(1.0+dd1))*(rho(il,j,n)-rho(ilm1,j,n)
      c                    -(p(il,j)-p(ilm1,j))*rccsq)
       r2=-(dd2/(1.0+dd2))*(p(il,j)-p(ilm1,j)
      c              +rr*cc*(sxp*u(il,j)+syp*v(il,j)-up))
```

```
      r3=-(dd1/(1.0+dd1))*(-syp*u(il,j)+sxp*v(il,j)-vp)
      r4=-(dd4/(1.0+dd4))*(p(il,j)-p(ilm1,j)
     c              -rr*cc*(sxp*u(il,j)+syp*v(il,j)-up))
      dp=0.5*(r2+r4)
      if (up.lt.cc) dp=0.0
      dup=(r2-dp)/(rr*cc)
      dvp=r3
      rm=rho(il,j,n)+r1+dp*rccsq
      um=u(il,j)+szp*dup-srp*dvp
      vm=v(il,j)+srp*dup+szp*dvp
      du(il,j,1)=rm-rho(il,j,n)
      du(il,j,2)=rm*um-rhou(il,j,n)
      du(il,j,3)=rm*vm-rhov(il,j,n)
      du(il,j,4)=(p(il,j)+dp)/gm1+0.5*rm*(um*um+vm*vm)-e(il,j,n)
   20 continue
      return
      end
c
c-----------------------------------------------------------------------
      subroutine bc
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      common/e/g11(101),g12(101),g13(101),g14(101),
     c         g21(101),g22(101),g23(101),g24(101),
     c         g31(101),g32(101),g33(101),g34(101),
     c         g41(101),g42(101),g43(101),g44(101),
     c         ff(101),implt



      gp1=gamma+1.0
c          ** bc for inlet **
      do 10 j=2,jlm1
      rtt=ptt/(cv*gm1*ttt)
      astr2=(2.0/gp1)*ggm1*cv*ttt
      um=u(1,j)
      v(1,j)=0.0
      rho(1,j,m)=rtt*(1.0-(gm1/gp1)*um*um/astr2)**(1.0/gm1)
      ei(1,j)=cv*ttt*(1.0-(gm1/gp1)*um*um/astr2)
      p(1,j)=gm1*rho(1,j,m)*ei(1,j)
      rhou(1,j,m)=rho(1,j,m)*um
      rhov(1,j,m)=0.0
      e(1,j,m)=rho(1,j,m)*(ei(1,j)+0.5*um*um)
   10 continue
```

```
c           ** bc for exit **
      pe=p(il,jl)
      iflag=0
      do 20 jz=2,jlm1
      j=jlm1+2-jz
      si=sqrt((z(il,j+1)-z(il,j))**2+(r(il,j+1)-r(il,j))**2)
      szp= (r(il,j+1)-r(il,j))/si
      srp=-(z(il,j+1)-z(il,j))/si
      up=szp*u(ilm1,j)+srp*v(ilm1,j)
      upil=szp*u(il,j)+srp*v(il,j)
      cc=sqrt(ggm1*ei(ilm1,j))
      ccil=sqrt(ggm1*ei(il,j))
      if (up.gt.cc.and.upil.gt.ccil) then
      iflag=1
      go to 20
      endif
      if (iflag.eq.1) go to 15
      dr=(pe-p(il,j))/(cc*cc)
      p(il,j)=pe
      rho(il,j,m)=rho(il,j,m)+dr
      rhou(il,j,m)=rho(il,j,m)*u(il,j)
      rhov(il,j,m)=rho(il,j,m)*v(il,j)
      ei(il,j)=p(il,j)/(gm1*rho(il,j,m))
      e(il,j,m)=rho(il,j,m)*(ei(il,j)+0.5*(u(il,j)**2+v(il,j)**2))
      go to 20
   15 continue
      rho(il,j,m)=rho(ilm1,j,m)
      rhou(il,j,m)=rhou(ilm1,j,m)
      rhov(il,j,m)=rhov(ilm1,j,m)
      e(il,j,m)=e(ilm1,j,m)
      u(il,j)=u(ilm1,j)
      v(il,j)=v(ilm1,j)
      p(il,j)=p(ilm1,j)
      ei(il,j)=ei(ilm1,j)
   20 continue
c
c     lower wall and centerline
c
      do 30 i=1,il
      u(i,1)=-u(i,2)
      v(i,1)=-v(i,2)
      ei(i,1)=ei(i,2)
     if(iadbwl.eq.0) ei(i,1)=2.0*eiwall-ei(i,2)
      p(i,1)=p(i,2)
      rho(i,1,m)=rho(i,2,m)*ei(i,2)/ei(i,1)
      u(i,jl)=u(i,jlm1)
      v(i,jl)=-v(i,jlm1)
      ei(i,jl)=ei(i,jlm1)
      p(i,jl)=p(i,jlm1)
      rho(i,jl,m)=rho(i,jlm1,m)
   30 continue
c
```

```
      return
      end
c-----------------------------------------------------------------------
      subroutine fsi(i,j,ii)
c-----------------------------------------------------------------------
c
     common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c        e(100,100,2),fs1(2),fs2(2),
     c        u(100,100),v(100,100),ei(100,100),p(100,100),
     c        fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c        gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c        il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c        iadbwl,eiwall,lcnt,cfl,
     c        nadv,nend,nvisc
      common/b/iturb,prt,ep(100,100)
      common/e/g11(101),g12(101),g13(101),g14(101),
     c         g21(101),g22(101),g23(101),g24(101),
     c         g31(101),g32(101),g33(101),g34(101),
     c         g41(101),g42(101),g43(101),g44(101),
     c         ff(101),implt
      common/splt/iflxs,sz,sr
      common/rad/nradq,rabsco
c
      aa=0.80
      bb=-7.10e-5
      sz=+(r(i+1,j+1)-r(i+1,j))
      sr=-(z(i+1,j+1)-z(i+1,j))
      qs=sz*u(ii,j)+sr*v(ii,j)
      qsi=sz*u(i,j)+sr*v(i,j)
      qsip1=sz*u(i+1,j)+sr*v(i+1,j)
      if (qsip1.gt.qsi) qs=0.5*(qsi+qsip1)
      piij=p(ii,j)
      fs1(k2)=rho(ii,j,n)*qs
      fs2(k2)=rhou(ii,j,n)*qs+piij*sz
      fs3(k2)=rhov(ii,j,n)*qs+piij*sr
      fs4(k2)=(e(ii,j,n)+piij)*qs
      if (iflxs.gt.0) call fsifs(i,j,ii)
      if (nadv.lt.nvisc) go to 11
      uiij=u(ii,j)
      viij=v(ii,j)
      due=u(i+1,j)-u(i,j)
      dve=v(i+1,j)-v(i,j)
      dee=ei(i+1,j)-ei(i,j)
      dze=0.25*(z(i+2,j+1)+z(i+2,j)-z(i,j+1)-z(i,j))
      dre=0.25*(r(i+2,j+1)+r(i+2,j)-r(i,j+1)-r(i,j))
      dun=0.5*(u(ii,j+1)-u(ii,j-1))
      dvn=0.5*(v(ii,j+1)-v(ii,j-1))
      den=0.5*(ei(ii,j+1)-ei(ii,j-1))
      dzn=r(i+1,j+1)-z(i+1,j)
      drn=r(i+1,j+1)-r(i+1,j)
      dzr=1.0/(dze*drn-dzn*dre)
      duz=(due*drn-dun*dre)*dzr
```

```
      dvz=(dve*drn-dvn*dre)*dzr
      dez=(dee*drn-den*dre)*dzr
      dur=-(due*dzn-dun*dze)*dzr
      dvr=-(dve*dzn-dvn*dze)*dzr
      der=-(dee*dzn-den*dze)*dzr
      temp=abs(0.5*(ei(i,j)+ei(i+1,j))/cv)
c     - use sutherland's form for air in mks unit-
c     - rmu=1.458e-6*sqrt(temp**3)/(temp+110.4)
c     - uf4 viscosity at 2000 K.
c    rmu=3.357e-6*sqrt(temp)/(aa+bb*temp)
     rmu=8.67e-5
c     - conductivity.
c
        rk=gamma*rmu/pr
c     - consider thermal radiation heat transfer-

      if(nradq.eq.1)then
c      consider radiative conductivity-
       sigma=5.67e-8
       tt=ei(i,j)/cv
       radk=(16.*sigma*(tt**3))/(3.*rabsco*cv)
c      rabsco-rosseland absorption coefficient-
       rk=rk+radk/cv
       endif
c
c
       rlmbda=-(2.0/3.0)*rmu
       stress=-rlmbda*(duz+dvr)
       sigz=stress-2.0*rmu*duz
       sigr=stress-2.0*rmu*dvr
       tau=-rmu*(dur+dvz)
       qz=rk*dez
       qr=rk*der
   11  continue
       return
       end

c-----------------------------------------------------------------------
      subroutine fsj(i,j,jj)
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c        e(100,100,2),fsl(2),fs2(2),
     c        u(100,100),v(100,100),ei(100,100),p(100,100),
     c        fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c        gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c        il,jl,ilm1,jlm1,jlfm,n,m,kl,k2,
     c        iadbwl,eiwall,lcnt,cfl,
     c        nadv,nend,nvisc
      common/b/iturb,prt,ep(100,100)
      common/c/du(100,100,4),dus(100,100,4)
      common/e/g11(101),g12(101),g13(101),g14(101),
```

```
c          g21(101),g22(101),g23(101),g24(101),
c          g31(101),g32(101),g33(101),g34(101),
c          g41(101),g42(101),g43(101),g44(101),
c          ff(101),implt
      common/splt/iflxs,sz,sr
      common/rad/nradq,rabsco

      aa=0.80
      bb=-7.10e-5
      sz=-(r(i+1,j+1)-r(i,j+1))
      sr=+(z(i+1,j+1)-z(i,j+1))
      qs=sz*u(i,jj)+sr*v(i,jj)
      qsj=sz*u(i,j)+sr*v(i,j)
      qsjp1=sz*u(i,j+1)+sr*v(i,j+1)
      if (qsjp1.gt.qsj) qs=0.5*(qsj+qsjp1)
      pijj=p(i,jj)
      if (j.eq.1.or.j.eq.jlm1) qs=0.0
      fs1(k2)=rho(i,jj,n)*qs
      fs2(k2)=rhou(i,jj,n)*qs+pijj*sz
      fs3(k2)=rhov(i,jj,n)*qs+pijj*sr
      fs4(k2)=(e(i,jj,n)+pijj)*qs
      if (iflxs.gt.0) call fsjfs(i,j,jj)
      if (nadv.lt.nvisc) go to 11
      uijj=0.5*(u(i,j)+u(i,j+1))
      vijj=0.5*(v(i,j)+v(i,j+1))
      due=0.5*(u(i+1,jj)-u(i-1,jj))
      dve=0.5*(v(i+1,jj)-v(i-1,jj))
      dee=0.5*(ei(i+1,jj)-ei(i-1,jj))
      dze=z(i+1,j+1)-z(i,j+1)
      dre=r(i+1,j+1)-r(i,j+1)
      dun=u(i,j+1)-u(i,j)
      dvn=v(i,j+1)-v(i,j)
      den=ei(i,j+1)-ei(i,j)
      dzn=0.25*(z(i+1,j+2)+z(i,j+2)-z(i+1,j)-z(i,j))
      drn=0.25*(r(i+1,j+2)+r(i,j+2)-r(i+1,j)-r(i,j))
      dzr=1.0/(dze*drn-dzn*dre)
      duz=(due*drn-dun*dre)*dzr
      dvz=(dve*drn-dvn*dre)*dzr
      dez=(dee*drn-den*dre)*dzr
      dur=-(due*dzn-dun*dze)*dzr
      dvr=-(dve*dzn-dvn*dze)*dzr
      der=-(dee*dzn-den*dze)*dzr
      temp=abs(0.5*(ei(i,j)+ei(i,j+1))/cv)
c     -Sutherland's form for air viscosity in mks unit-
c     rmu=1.458e-06*sqrt(temp**3)/(temp+110.4)
c     - uf4 viscosity at 2000 K
c     rmu=3.357e-6*sqrt(temp)/(aa+bb*temp)
      rmu=8.67e-5
c
      rk=gamma*rmu/pr
      rmu=rmu+ep(i,j)
      rk=rk+gamma*ep(i,j)/prt
```

```
c
c      - consider radiation heat transfer-
       if(nradq.eq.1)then
c      - consider radiative conductivity-
        sigma=5.67e-8
c      - rabsco-rosseland absorption coefficient-
        tt=ei(i,j)/cv
        radk=(16.*sigma*(tt**3))/(3.*rabsco*cv)
        rk=rk+radk/cv
       endif
c
       rlmbda=-(2.0/3.0)*rmu
       stress=-rlmbda*(duz+dvr)
       sigz=stress-2.0*rmu*duz
       sigr=stress-2.0*rmu*dvr
       tau=-rmu*(dur+dvz)
       qz=rk*dez
       qr=rk*der
       fs2(k2)=fs2(k2)+sigz*sz+tau*sr
       fs3(k2)=fs3(k2)+tau*sz+sigr*sr
       fs4(k2)=fs4(k2)+(sigz*uijj+tau*vijj-qz)*sz+
      c                (tau*uijj+sigr*vijj-qr)*sr
    11 continue
       return
       end
c
c-----------------------------------------------------------------------
       subroutine fsh(i,j,jj)
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
      c        e(100,100,2),fs1(2),fs2(2),
      c        u(100,100),v(100,100),ei(100,100),p(100,100),
      c        fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
      c        gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
      c        il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
      c        iadbwl,eiwall,lcnt,cfl,
      c        nadv,nend,nvisc
       common/b/iturb,prt,ep(100,100)
       common/c/du(100,100,4),dus(100,100,4)
       common/e/g11(101),g12(101),g13(101),g14(101),
      c         g21(101),g22(101),g23(101),g24(101),
      c         g31(101),g32(101),g33(101),g34(101),
      c         g41(101),g42(101),g43(101),g44(101),
      c         ff(101),implt
       common/splt/iflxs,sz,sr
       common/q/qp
       common/rad/nradq,rabsco

       aa=0.80
       bb=-7.10e-5
       sz=-(r(i+1,j+1)-r(i,j+1))
```

```
      sr=+(z(i+1,j+1)-z(i,j+1))
      qs=sz*u(i,jj)+sr*v(i,jj)
      qsj=sz*u(i,j)+sr*v(i,j)
      qsjp1=sz*u(i,j+1)+sr*v(i,j+1)
      if (qsjp1.gt.qsj) qs=0.5*(qsj+qsjp1)
      pijj=p(i,jj)
      if (j.eq.1.or.j.eq.jlm1) qs=0.0
      fs1(k2)=-rho(i,jj,n)*qs
      fs2(k2)=-rhou(i,jj,n)*qs
      fs3(k2)=-rhov(i,jj,n)*qs
      fs4(k2)=-(e(i,jj,n)+pijj)*qs
      if (iflxs.gt.0) call fsjfs(i,j,jj)
      if (nadv.lt.nvisc) go to 11
      uijj=0.5*(u(i,j)+u(i,j+1))
      vijj=0.5*(v(i,j)+v(i,j+1))
c
      ry=0.5*(r(i,j)+r(i,j+1))
      due=0.5*(u(i+1,jj)-u(i-1,jj))
      dve=0.5*(v(i+1,jj)-v(i-1,jj))
      dee=0.5*(ei(i+1,jj)-ei(i-1,jj))
      dze=z(i+1,j+1)-z(i,j+1)
      dre=r(i+1,j+1)-r(i,j+1)
      dun=u(i,j+1)-u(i,j)
      dvn=v(i,j+1)-v(i,j)
      den=ei(i,j+1)-ei(i,j)
      dzn=0.25*(z(i+1,j+2)+z(i,j+2)-z(i+1,j)-z(i,j))
      drn=0.25*(r(i+1,j+2)+r(i,j+2)-r(i+1,j)-r(i,j))
      dzr=1.0/(dze*drn-dzn*dre)
      duz=(due*drn-dun*dre)*dzr
      dvz=(dve*drn-dvn*dre)*dzr
      dez=(dee*drn-den*dre)*dzr
      dur=-(due*dzn-dun*dze)*dzr
      dvr=-(dve*dzn-dvn*dze)*dzr
      der=-(dee*dzn-den*dze)*dzr
      temp=abs(0.5*(ei(i,j)+ei(i,j+1))/cv)
c     - uf4 viscosity at 2000 K
      rmu=8.67e-5
c     rmu=3.357e-6*sqrt(temp)/(aa+bb*temp)
c    - viscosity for air.
c
c       rmu=1.458e-6*sqrt(temp**3)/(temp+110.4)
        rk=gamma*rmu/pr
        rmu=rmu+ep(i,j)
        rk=rk+gamma*ep(i,j)/prt
c    - consider radiation heat transfer-
         if(nradq.eq.1)then
c    - consider radiative conductivity-
         sigma=5.67e-8
c       rabsco-rosseland absorption coefficient-
         tt=ei(i,j)/cv
         radk=(16.*sigma*(tt**3))/(3.*rabsco*cv)
         rk=rk+radk/cv
```

```
        endif
c
c
      rlmbda=-(2.0/3.0)*rmu
      stress=-rlmbda*(duz+dvr)
      sigz=stress-2.0*rmu*duz
      sigr=stress-2.0*rmu*dvr
      tau=-rmu*(dur+dvz)
      qz=rk*dez
      qr=rk*der
      fs2(k2)=fs2(k2)+sigz*sz+tau*sr
      fs3(k2)=fs3(k2)+tau*sz+sigr*sr+stress
     c        -(4./3.)*rmu*uijj/ry
      fs4(k2)=fs4(k2)+(sigz*uijj+tau*vijj-qz)*sz+
     c                (tau*uijj+sigr*vijj-qr)*sr
   11 continue
      return
      end

c-----------------------------------------------------------------------
      subroutine l(dt)
c-----------------------------------------------------------------------

      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c        e(100,100,2),fs1(2),fs2(2),
     c        u(100,100),v(100,100),ei(100,100),p(100,100),
     c        fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c        gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c        il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c        iadbwl,eiwall,lcnt,cfl,
     c        nadv,nend,nvisc
      common/c/du(100,100,4),dus(100,100,4)
      common/e/g11(101),g12(101),g13(101),g14(101),
     c         g21(101),g22(101),g23(101),g24(101),
     c         g31(101),g32(101),g33(101),g34(101),
     c         g41(101),g42(101),g43(101),g44(101),
     c         ff(101),implt
      common/splt/iflxs,sz,sr
      common/tem/t(101,101),ttwall
      common/q/qp

      lcnt=lcnt+1
      iadd=mod(nadv,2)
      jadd=mod(((iadd+nadv)/2),2)
      do 20 n=1,2
      m=3-n
      nm1=n-1
      call lbc(dt)
      do 5 j=2,jlm1
      call fsi(l,j,1+iadd)
      do 5 i=2,ilm1
      k=k1
```

```
      k1=k2
      k2=k
      call fsi(i,j,i+iadd)
      dtv=dt/vol(i,j)
      du(i,j,1)=-dtv*(fs1(k2)-fs1(k1))
      du(i,j,2)=-dtv*(fs2(k2)-fs2(k1))
      du(i,j,3)=-dtv*(fs3(k2)-fs3(k1))
      du(i,j,4)=-dtv*(fs4(k2)-fs4(k1))
    5 continue
      do 10 i=2,ilm1
      call fsj(i,1,1+jadd)
      do 10 j=2,jlm1
      k=k1
      k1=k2
      k2=k
      call fsj(i,j,j+jadd)
      dtv=dt/vol(i,j)
      du(i,j,1)=du(i,j,1)-dtv*(fs1(k2)-fs1(k1))
      du(i,j,2)=du(i,j,2)-dtv*(fs2(k2)-fs2(k1))
      du(i,j,3)=du(i,j,3)-dtv*(fs3(k2)-fs3(k1))
      du(i,j,4)=du(i,j,4)-dtv*(fs4(k2)-fs4(k1))
   10 continue
c
      do 11 i = 2,ilm1
      call fsh(i,1,1+jadd)
      do 11 j = 2,jlm1
      call fsh(i,j,j+jadd)
      dtv=dt/vol(i,j)
      du(i,j,1)=du(i,j,1)+dtv*(fs1(k2)*vol(i,j))
      du(i,j,2)=du(i,j,2)+dtv*(fs2(k2)*vol(i,j))
      du(i,j,3)=du(i,j,3)+dtv*(fs3(k2)*vol(i,j))
      du(i,j,4)=du(i,j,4)+dtv*(fs4(k2)*vol(i,j))+qp*dt
   11 continue
c
      if (implt.gt.0) call limplt(dt,iadd,jadd)
      rn=n
      rnm1=rn-1.0
      do 12 i=1,il
      do 12 j=2,jlm1
      rho(i,j,m)=(rnm1*rho(i,j,m)+rho(i,j,n)+du(i,j,1))/rn
      rhou(i,j,m)=(rnm1*rhou(i,j,m)+rhou(i,j,n)+du(i,j,2))/rn
      rhov(i,j,m)=(rnm1*rhov(i,j,m)+rhov(i,j,n)+du(i,j,3))/rn
      e(i,j,m)=(rnm1*e(i,j,m)+e(i,j,n)+du(i,j,4))/rn
      u(i,j)=rhou(i,j,m)/rho(i,j,m)
      v(i,j)=rhov(i,j,m)/rho(i,j,m)
      ei(i,j)=e(i,j,m)/rho(i,j,m)-0.5*(u(i,j)*u(i,j)+v(i,j)*v(i,j))
      p(i,j)=gm1*rho(i,j,m)*ei(i,j)
c
      t(i,j)=ei(i,j)/cv
c
   12 continue
      call bc
```

```
      iadd=mod((iadd+1),2)
      jadd=mod((jadd+1),2)
   20 continue
      return
      end
c
c
c    if 1-D integral approximation is used, this subroutine, rq,
c    must be included.
c-----------------------------------------------------------------
      subroutine rq
c-----------------------------------------------------------------
c     common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
c     c        e(100,100,2),fs1(2),fs2(2),
c     c        u(100,100),v(100,100),ei(100,100),p(100,100),
c     c        fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
c     c        gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
c     c        il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
c     c        iadbwl,eiwall,lcnt,cfl,
c     c        nadv,nend,nvisc
c      common/c/du(100,100,4),dus(100,100,4)
c      common/e/g11(101),g12(101),g13(101),g14(101),
c     c         g21(101),g22(101),g23(101),g24(101),
c     c         g31(101),g32(101),g33(101),g34(101),
c     c         g41(101),g42(101),g43(101),g44(101),
c     c         ff(101),implt
c      common/splt/iflxs,sz,sr
c      common/tem/t(101,101), ttwall
c      common/q/qp
c      common/r/radq(100,100)
c      common/rad/nradq, rabsco
c      dimension rr(100), plusi(100,100), rminusi(100,100)
c
c      calculate + direction radiaitve intensity
c      pi=3.14159
c      sigma=5.67e-8
c      do 1 i =2,il-1
c      do 2 k =2,jl-1
c      r=0.0
c      plusin=0.0
c      do 3 j =2,jl-2
c      rr(j)=0.5*(r(i,j+2)-r(i,j))
c      r=r+rr(j)
c      if(r.le.(5./rabsco))then
c      ri=((t(i,j)+t(i,j+1))*0.5)**4
c      plusi(i,j)=rabsco*sigma*ri*(1./pi)*exp(-rabsco*rr(j))*rr(j)
c      plusin=plusin+plusi(i,j)
c      endif

c3     continue
c      plusi(i,k)=plusin*2.*pi*r(i,k+1)*(z(i+1,k)-z(i,k))
c2     continue
```

```
c
c     - calculate - direction radiative intensity
c       aaw=rabsco*(ttt/ttwall)
c       do 4 k=jl,4,-1
c       rminusin=0.0
c       r=0.0
c       do 11 j=jl,4,-1
c       rw=r(i,j)-r(i,2)
c       rr(j)=0.5*(r(i,j)-r(i,j-2))
c       r=r+rr(j)
c       if(r.le.(5./rabsco). and. rw.gt.(5./aaw))then
c       ri=((t(i,j)+t(i,j-1))*0.5)**4
c       rminusi(i,j)=rabsco*sigma*ri*(1./pi)*
c     *  exp(-rabsco*rr(j))*rr(j)
c       rminusin=rminusin+rminusi(i,j)
c       endif
c11     continue
c       rminusi(i,k)=rminusin
c       if(r.le.(5./rabsco). and. rw.le.(5./aaw)then
c       rminusi(i,k)=rminuin+(1./pi)*sigma*(ttwall**4)*exp(-rw*aaw)
c       endif
c4      continue
c1      continue
c     - calculate radiative energy deposited in every cell.
c
c       do 12 i = 2, il-1
c       do 12 k = 2, jl-2
c       radq(i,k)=plusi(i,k)-rminusi(i,k)
c12     continue
c       return
c       end
c
c
c-----------------------------------------------------------------------
      subroutine impli(dt,i,j,iadd)
c-----------------------------------------------------------------------
c
c     -implicit procedure in i- direction.
       common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
      c         e(100,100,2),fs1(2),fs2(2),
      c         u(100,100),v(100,100),ei(100,100),p(100,100),
      c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
      c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
      c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
      c         iadbwl,eiwall,lcnt,cfl,
      c         nadv,nend,nvisc
       common/b/iturb,prt,ep(100,100)
       common/c/du(100,100,4),dus(100,100,4)
       common/d/a11,a12,a13,a14,a21,a22,a23,a24,ffi(2),
      c          a31,a32,a33,a34,a41,a42,a43,a44,ni
       common/e/g11(101),g12(101),g13(101),g14(101),
      c         g21(101),g22(101),g23(101),g24(101),
```

```
c          g31(101),g32(101),g33(101),g34(101),
c          g41(101),g42(101),g43(101),g44(101),
c          ff(101),implt
      common/splt/iflxs,sz,sr
c
c

      beta=gamma-1.0
      sign=1.0
      do 20 nn=1,2
      ii=i+2-nn
      si=sqrt((z(ii,j+1)-z(ii,j))**2+(r(ii,j+1)-r(ii,j))**2)
      szp= (r(ii,j+1)-r(ii,j))/si
      srp=-(z(ii,j+1)-z(ii,j))/si
      ii=ii-1+iadd
c      if (ii.eq.il) ii=ii-1
      rr=p(ii,j)/(gm1*ei(ii,j))
      uu=u(ii,j)
      vv=v(ii,j)
      cc=sqrt(ggm1*abs(ei(ii,j)))
      up= szp*uu+srp*vv
      vp=-srp*uu+szp*vv
      alp=0.5*(uu*uu+vv*vv)
      rccsq=1.0/(cc*cc)
      dtvol=dt/vol(i,j)
      iflag=1
      if (ni.eq.2.and.nn.eq.1) iflag=2
    6 continue
      il=i+2-nn+1-iflag
      i2=il-3+2*iflag
      aa=1.0
      if (iflxs.eq.2) aa=1.5
      dd=-aa*sign*(3-2*(iflag))*si*dtvol
      dd0=0.0
      if (abs(p(il,j)-p(i2,j)).gt.
     c    0.5*amin1(abs(p(il,j)),abs(p(i2,j)))) then
      rr=rho(il,j,n)
      uu=u(il,j)
      vv=v(il,j)
      up= szp*uu+srp*vv
      vp=-srp*uu+szp*vv
      cc=sqrt(ggm1*abs(ei(il,j)))
      alp=0.5*(up*up+vp*vp)
      dd0=0.5*abs(dd)*amax1(cc-abs(up),0.0)/gamma
      endif

      dd1=up*dd
      dd2=(up+cc)*dd
      dd4=(up-cc)*dd
      il=i+2-nn

      coef=1.0
      if (il.eq.2.or.il.eq.il) coef=0.0
```

```
      ep1=coef*dd*(szp*(u(i1,j)-u(i1-1,j))+srp*(v(i1,j)-v(i1-1,j)))
      ep2=coef*dd*(szp*(u(i1,j)-u(i1-1,j))+srp*(v(i1,j)-v(i1-1,j))
     c +(sqrt(ggm1*abs(ei(i1,j)))-sqrt(ggm1*abs(ei(i1-1,j)))))
      ep4=coef*dd*(sxp*(u(i1,j)-u(i1-1,j))+syp*(v(i1,j)-v(i1-1,j))
     c -(sqrt(ggm1*abs(ei(i1,j)))-sqrt(ggm1*abs(ei(i1-1,j)))))

      dd1=0.5*(dd1+sqrt(dd1*dd1+ep1*ep1))+dd0
      dd2=0.5*(dd2+sqrt(dd2*dd2+ep2*ep2))+dd0
      dd4=0.5*(dd4+sqrt(dd4*dd4+ep4*ep4))+dd0

      i1=i+2-nn+1-iflag
      t11= dd1*(1.0-alp*beta*rccsq)
      t12= dd1*uu*beta*rccsq
      t13= dd1*vv*beta*rccsq
      t14=-dd1*beta*rccsq
      t21= dd2*(alp*beta-up*cc)
      t22= dd2*( sxp*cc-uu*beta)
      t23= dd2*( syp*cc-vv*beta)
      t24= dd2*beta
      t31=-dd1*vp/rr
      t32=-dd1*syp/rr
      t33= dd1*sxp/rr
      t34= 0.0
      t41= dd4*(alp*beta+up*cc)
      t42= dd4*(-szp*cc-uu*beta)
      t43= dd4*(-srp*cc-vv*beta)
      t44= dd4*beta
c
      if (iflag.eq.2) go to 8
      bp11= 1.0
      bp12= 0.5*rccsq
      bp13= 0.0
      bp14= 0.5*rccsq
      bp21= uu
      bp22= 0.5*(uu+szp*cc)*rccsq
      bp23=-srp*rr
      bp24= 0.5*(uu-szp*cc)*rccsq
      bp31= vv
      bp32= 0.5*(vv+srp*cc)*rccsq
      bp33= szp*rr
      bp34= 0.5*(vv-srp*cc)*rccsq
      bp41= alp
      bp42= 0.5*((alp+up*cc)*rccsq+1.0/beta)
      bp43= rr*vp
      bp44= 0.5*((alp-up*cc)*rccsq+1.0/beta)
      b11=-(bp11*t11+bp12*t21+bp13*t31+bp14*t41)
      b12=-(bp11*t12+bp12*t22+bp13*t32+bp14*t42)
      b13=-(bp11*t13+bp12*t23+bp13*t33+bp14*t43)
      b14=-(bp11*t14+bp12*t24+bp13*t34+bp14*t44)
      b21=-(bp21*t11+bp22*t21+bp23*t31+bp24*t41)
      b22=-(bp21*t12+bp22*t22+bp23*t32+bp24*t42)
      b23=-(bp21*t13+bp22*t23+bp23*t33+bp24*t43)
```

```
      b24=-(bp21*t14+bp22*t24+bp23*t34+bp24*t44)
      b31=-(bp31*t11+bp32*t21+bp33*t31+bp34*t41)
      b32=-(bp31*t12+bp32*t22+bp33*t32+bp34*t42)
      b33=-(bp31*t13+bp32*t23+bp33*t33+bp34*t43)
      b34=-(bp31*t14+bp32*t24+bp33*t34+bp34*t44)
      b41=-(bp41*t11+bp42*t21+bp43*t31+bp44*t41)
      b42=-(bp41*t12+bp42*t22+bp43*t32+bp44*t42)
      b43=-(bp41*t13+bp42*t23+bp43*t33+bp44*t43)
      b44=-(bp41*t14+bp42*t24+bp43*t34+bp44*t44)
      iflag=2
      ab=1.0
      ac=0.0
      if (nn.eq.1) go to 12
      go to 14
    8 continue
      cp11= 1.0
      cp12= 0.5*rccsq
      cp13= 0.0
      cp14= 0.5*rccsq
      cp21= uu
      cp22= 0.5*(uu+szp*cc)*rccsq
      cp23=-syp*rr
      cp24= 0.5*(uu-szp*cc)*rccsq
      cp31= vv
      cp32= 0.5*(vv+srp*cc)*rccsq
      cp33= szp*rr
      cp34= 0.5*(vv-srp*cc)*rccsq
      cp41= alp
      cp42= 0.5*((alp+up*cc)*rccsq+1.0/beta)
      cp43= rr*vp
      cp44= 0.5*((alp-up*cc)*rccsq+1.0/beta)
      c11=-(cp11*t11+cp12*t21+cp13*t31+cp14*t41)
      c12=-(cp11*t12+cp12*t22+cp13*t32+cp14*t42)
      c13=-(cp11*t13+cp12*t23+cp13*t33+cp14*t43)
      c14=-(cp11*t14+cp12*t24+cp13*t34+cp14*t44)
      c21=-(cp21*t11+cp22*t21+cp23*t31+cp24*t41)
      c22=-(cp21*t12+cp22*t22+cp23*t32+cp24*t42)
      c23=-(cp21*t13+cp22*t23+cp23*t33+cp24*t43)
      c24=-(cp21*t14+cp22*t24+cp23*t34+cp24*t44)
      c31=-(cp31*t11+cp32*t21+cp33*t31+cp34*t41)
      c32=-(cp31*t12+cp32*t22+cp33*t32+cp34*t42)
      c33=-(cp31*t13+cp32*t23+cp33*t33+cp34*t43)
      c34=-(cp31*t14+cp32*t24+cp33*t34+cp34*t44)
      c41=-(cp41*t11+cp42*t21+cp43*t31+cp44*t41)
      c42=-(cp41*t12+cp42*t22+cp43*t32+cp44*t42)
      c43=-(cp41*t13+cp42*t23+cp43*t33+cp44*t43)
      c44=-(cp41*t14+cp42*t24+cp43*t34+cp44*t44)
      ab=0.0
      ac=1.0
      if (nn.eq.1) go to 14
      if (ni.eq.1) go to 20
      i1=i-1
```

```
      du1=du(i1,j,1)
      du2=du(i1,j,2)
      du3=du(i1,j,3)
      du4=du(i1,j,4)
      du(i,j,1)=du(i,j,1)-(c11*du1+c12*du2+c13*du3+c14*du4)
      du(i,j,2)=du(i,j,2)-(c21*du1+c22*du2+c23*du3+c24*du4)
      du(i,j,3)=du(i,j,3)-(c31*du1+c32*du2+c33*du3+c34*du4)
      du(i,j,4)=du(i,j,4)-(c41*du1+c42*du2+c43*du3+c44*du4)
      go to 20
   12 continue
      i1=i+1
      du1=du(i1,j,1)
      du2=du(i1,j,2)
      du3=du(i1,j,3)
      du4=du(i1,j,4)
      bdu1=b11*du1+b12*du2+b13*du3+b14*du4
      bdu2=b21*du1+b22*du2+b23*du3+b24*du4
      bdu3=b31*du1+b32*du2+b33*du3+b34*du4
      bdu4=b41*du1+b42*du2+b43*du3+b44*du4
      du(i,j,1)=du(i,j,1)-bdu1
      du(i,j,2)=du(i,j,2)-bdu2
      du(i,j,3)=du(i,j,3)-bdu3
      du(i,j,4)=du(i,j,4)-bdu4
      dus(i,j,1)=dus(i,j,1)-bdu1
      dus(i,j,2)=dus(i,j,2)-bdu2
      dus(i,j,3)=dus(i,j,3)-bdu3
      dus(i,j,4)=dus(i,j,4)-bdu4
      go to 6
   14 continue
      a11=a11-ab*b11-ac*c11
      a12=a12-ab*b12-ac*c12
      a13=a13-ab*b13-ac*c13
      a14=a14-ab*b14-ac*c14
      a21=a21-ab*b21-ac*c21
      a22=a22-ab*b22-ac*c22
      a23=a23-ab*b23-ac*c23
      a24=a24-ab*b24-ac*c24
      a31=a31-ab*b31-ac*c31
      a32=a32-ab*b32-ac*c32
      a33=a33-ab*b33-ac*c33
      a34=a34-ab*b34-ac*c34
      a41=a41-ab*b41-ac*c41
      a42=a42-ab*b42-ac*c42
      a43=a43-ab*b43-ac*c43
      a44=a44-ab*b44-ac*c44
      if (nn.eq.2) go to 6
   20 continue
      return
      end
```

```
c-----------------------------------------------------------------
      subroutine limplt(dt,iadd,jadd)
c-----------------------------------------------------------------
c
c**** implicit procedure in j- direction.
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      common/b/iturb,prt,ep(100,100)
      common/c/du(100,100,4),dus(100,100,4)
      common/d/a11,a12,a13,a14,a21,a22,a23,a24,ffi(2),
     c         a31,a32,a33,a34,a41,a42,a43,a44,ni
      common/e/g11(101),g12(101),g13(101),g14(101),
     c         g21(101),g22(101),g23(101),g24(101),
     c         g31(101),g32(101),g33(101),g34(101),
     c         g41(101),g42(101),g43(101),g44(101),
     c         ff(101),implt
      common/splt/iflxs,sz,sr
      dimension dui(100,4)
      data a11,a12,a13,a14,a21,a22,a23,a24,a31,a32,a33,a34,
     c     a41,a42,a43,a44,b11,b12,b13,b14,b21,b22,b23,b24,
     c     b31,b32,b33,b34,b41,b42,b43,b44,c11,c12,c13,c14,
     c    c21,c22,c23,c24,c31,c32,c33,c34,c41,c42,c43,c44/48*0.0/
c
      beta= gamma-1.0
      sign=1.0
      aa=0.80
      bb=-7.10e-5
      bc=1.0/3.0
      do 1 i=1,il
      do 1 j=1,jl
      do 1 l=1,4
      dus(i,j,l)=du(i,j,l)
    1 continue
c      do 20 ni=1,3
      do 20 ni=1,2
      do 20 iz=2,ilm1
      i=ilm1+2-iz
      if (ni.eq.2) i=iz
      if (ni.eq.1) go to 3
      do 2 j=1,jl
      do 2 l=1,4
      dui(j,l)=du(i,j,l)
      du(i,j,l)=dus(i,j,l)
    2 continue
    3 continue
      ttu=1.0
```

```
      vol(i,jl+1)=vol(i,jl)
      do 10 jz=2,jl
      j=jl+2-jz
      sj=sqrt((z(i+1,j)-z(i,j))**2+(r(i+1,j)-r(i,j))**2)
      srp= (z(i+1,j)-z(i,j))/sj
      szp=-(r(i+1,j)-r(i,j))/sj
      vr=vol(i,j+1)/vol(i,j)
      temp=abs(0.5*(ei(i,j)+ei(i,j-1))/cv)
c     - use sutherland's form for air viscosity -
c     rmu=1.458e-06*sqrt(temp**3)/(temp+110.4)
c     - uf4 property
c     rmu=3.367e-6*sqrt(temp)/(aa+bb*temp)
c     UF4 viscosity at 2000 K
      rmu=8.67e-5
c
c
      if (nadv.lt.nvisc) rmu=0.0
      rk=gamma*rmu/pr
      rmu=rmu+ep(i,j-1)
      rk=rk+gamma*ep(i,j-1)/prt
      rlmbda=-(2.0/3.0)*rmu
      jj=j-1+jadd
      rr=p(i,jj)/(gm1*ei(i,jj))
      cc=sqrt(ggm1*abs(ei(i,jj)))
      rccsq=1.0/(cc*cc)
      upbar=0.5*(srp*(u(i,j)+u(i,j-1))-szp*(v(i,j)+v(i,j-1)))
      vpbar=0.5*(szp*(u(i,j)+u(i,j-1))+srp*(v(i,j)+v(i,j-1)))
      dtvol=dt/vol(i,j)
      iflag=1
      if (j.eq.jl) go to 6
      du1=du(i,j+1,1)
      du2=du(i,j+1,2)
      du3=du(i,j+1,3)
      du4=du(i,j+1,4)
      du(i,j,1)=du(i,j,1)-vr*(b11*du1+b12*du2+b13*du3+b14*du4)
      du(i,j,2)=du(i,j,2)-vr*(b21*du1+b22*du2+b23*du3+b24*du4)
      du(i,j,3)=du(i,j,3)-vr*(b31*du1+b32*du2+b33*du3+b34*du4)
      du(i,j,4)=du(i,j,4)-vr*(b41*du1+b42*du2+b43*du3+b44*du4)
      a11=b11*g11(j+1)+b12*g21(j+1)+b13*g31(j+1)+b14*g41(j+1)
      a12=b11*g12(j+1)+b12*g22(j+1)+b13*g32(j+1)+b14*g42(j+1)
      a13=b11*g13(j+1)+b12*g23(j+1)+b13*g33(j+1)+b14*g43(j+1)
      a14=b11*g14(j+1)+b12*g24(j+1)+b13*g34(j+1)+b14*g44(j+1)
      a21=b21*g11(j+1)+b22*g21(j+1)+b23*g31(j+1)+b24*g41(j+1)
      a22=b21*g12(j+1)+b22*g22(j+1)+b23*g32(j+1)+b24*g42(j+1)
      a23=b21*g13(j+1)+b22*g23(j+1)+b23*g33(j+1)+b24*g43(j+1)
      a24=b21*g14(j+1)+b22*g24(j+1)+b23*g34(j+1)+b24*g44(j+1)
      a31=b31*g11(j+1)+b32*g21(j+1)+b33*g31(j+1)+b34*g41(j+1)
      a32=b31*g12(j+1)+b32*g22(j+1)+b33*g32(j+1)+b34*g42(j+1)
      a33=b31*g13(j+1)+b32*g23(j+1)+b33*g33(j+1)+b34*g43(j+1)
      a34=b31*g14(j+1)+b32*g24(j+1)+b33*g34(j+1)+b34*g44(j+1)
      a41=b41*g11(j+1)+b42*g21(j+1)+b43*g31(j+1)+b44*g41(j+1)
      a42=b41*g12(j+1)+b42*g22(j+1)+b43*g32(j+1)+b44*g42(j+1)
```

```
      a43=b41*g13(j+1)+b42*g23(j+1)+b43*g33(j+1)+b44*g43(j+1)
      a44=b41*g14(j+1)+b42*g24(j+1)+b43*g34(j+1)+b44*g44(j+1)
    6 continue
      uu=u(i,jj)
      vv=v(i,jj)
      up= srp*uu-szp*vv
      vp= szp*uu+srp*vv
      alp=0.5*(up*up+vp*vp)
      j1=j+1-iflag
      rrj=p(i,j1)/(gm1*ei(i,j1))
      uuj=u(i,j1)
      vvj=v(i,j1)
      upj=srp*uuj-szp*vvj
      vpj=szp*uuj+srp*vvj
      ccj=sqrt(ggm1*abs(ei(i,j1)))
      alpj=0.5*(uuj*uuj+vvj*vvj)
      aa=1.0
      if (iflxs.eq.2) aa=1.5
      dd=-aa*sign*(3-2*(iflag))*sj*dtvol
      dd0=0.0
      if (abs(p(i,j)-p(i,j-1)).gt.
     c    0.5*amin1(abs(p(i,j)),abs(p(i,j-1)))) then
      rr=rho(i,j1,n)
      uu=u(i,j1)
      vv=v(i,j1)
      up= srp*uu-szp*vv
      vp= szp*uu+srp*vv
      cc=sqrt(ggm1*abs(ei(i,j1)))
      alp=0.5*(up*up+vp*vp)
      dd0=0.5*abs(dd)*amax1(cc-abs(vp),0.0)/gamma
      endif
      ttb=1.0
      ttc=1.0
      tt=0.0
      if (j.eq.2) then
      if (iflag.eq.2) go to 9
      dd=0.0
      ttc=0.0
      ttu=1.0
      tte=1-2*iadbwl
      tt=1.0
      endif
      if (j.eq.j1) then
      dd=0.0
      ttb=0.0
      ttu=-1.0
      tte=-1.0
      tt=1.0
      endif
      dd1=vp*dd
      dd3=(vp+cc)*dd
      dd4=(vp-cc)*dd
```

```
 coef=1.0

 ep1=coef*dd*(szp*(u(i,j)-u(i,j-1))+srp*(v(i,j)-v(i,j-1)))
 ep3=coef*dd*(szp*(u(i,j)-u(i,j-1))+srp*(v(i,j)-v(i,j-1))
c       +(sqrt(ggm1*abs(ei(i,j)))-sqrt(ggm1*abs(ei(i,j-1)))))
 ep4=coef*dd*(szp*(u(i,j)-u(i,j-1))+srp*(v(i,j)-v(i,j-1))
c       -(sqrt(ggm1*abs(ei(i,j)))-sqrt(ggm1*abs(ei(i,j-1)))))
 dd1=0.5*(dd1+sqrt(dd1*dd1+ep1*ep1))+dd0
 dd3=0.5*(dd3+sqrt(dd3*dd3+ep3*ep3))+dd0
 dd4=0.5*(dd4+sqrt(dd4*dd4+ep4*ep4))+dd0
 if (j.le.jlfm) then
 wt1=((z(i,j)-z(i,2))**2+(r(i,j)-r(i,2))**2)/
c     ((z(i,jlfm)-z(i,2))**2+(r(i,jlfm)-r(i,2))**2)
 wt2=1.0-wt1
 uu=wt1*uu+wt2*uuj
 vv=wt1*vv+wt2*vvj
 alp=0.5*(uu*uu+vv*vv)
 endif

 t11= dd1*(1.0-alp*beta*rccsq)
 t12= dd1*uu*beta*rccsq
 t13= dd1*vv*beta*rccsq
 t14=-dd1*beta*rccsq
 t21=-dd1*up/rr
 t22= dd1*srp/rr
 t23=-dd1*szp/rr
 t24= 0.0
 t31= dd3*(alp*beta-vp*cc)
 t32= dd3*( szp*cc-uu*beta)
 t33= dd3*( srp*cc-vv*beta)
 t34= dd3*beta
 t41= dd4*(alp*beta+vp*cc)
 t42= dd4*(-szp*cc-uu*beta)
 t43= dd4*(-srp*cc-vv*beta)
 t44= dd4*beta

 if (j.le.jlfm) then
 vp=wt1*vp+wt2*vpj
 uu= srp*up+szp*vp
 vv=-szp*up+srp*vp
 alp=0.5*(uu*uu+vv*vv)
 endif

 if (iflag.eq.2) go to 8
 bp11= 1.0
 bp12= 0.0
 bp13= 0.5*rccsq
 bp14= 0.5*rccsq
 bp21= uu
 bp22= srp*rr
 bp23= (uu+szp*cc)*0.5*rccsq
 bp24= (uu-szp*cc)*0.5*rccsq
```

```
      bp31= vv
      bp32=-szp*rr
      bp33= (vv+srp*cc)*0.5*rccsq
      bp34= (vv-srp*cc)*0.5*rccsq
      bp41= alp
      bp42= rr*up
      bp43= 0.5*((alp+vp*cc)*rccsq+1.0/beta)
      bp44= 0.5*((alp-vp*cc)*rccsq+1.0/beta)
      vdd=sj*dtvol*sj/(.5*rrj*(vol(i,j)+vol(i,j-1)))
      rnu1=ttb*(1.0+tt*ttu)*rmu
      rnu2=ttb*(1.0+tt)*(rlmbda+2.0*rmu)
      rnu3=ttb*(1.0+tt*tte)*rk
      b11=-(bp11*t11+bp12*t21+bp13*t31+bp14*t41)
      b12=-(bp11*t12+bp12*t22+bp13*t32+bp14*t42)
      b13=-(bp11*t13+bp12*t23+bp13*t33+bp14*t43)
      b14=-(bp11*t14+bp12*t24+bp13*t34+bp14*t44)
      b21=-(bp21*t11+bp22*t21+bp23*t31+bp24*t41)
     c       +vdd*( srp*upj*rnu1+szp*vpj*rnu2)
      b22=-(bp21*t12+bp22*t22+bp23*t32+bp24*t42)
     c       -vdd*(srp*srp*rnu1+szp*szp*rnu2)
      b23=-(bp21*t13+bp22*t23+bp23*t33+bp24*t43)
     c       +vdd*szp*srp*(rnu1-rnu2)
      b24=-(bp21*t14+bp22*t24+bp23*t34+bp24*t44)
      b31=-(bp31*t11+bp32*t21+bp33*t31+bp34*t41)
     c       +vdd*(-szp*upj*rnu1+srp*vpj*rnu2)
      b32=-(bp31*t12+bp32*t22+bp33*t32+bp34*t42)
     c       +vdd*szp*srp*(rnu1-rnu2)
      b33=-(bp31*t13+bp32*t23+bp33*t33+bp34*t43)
     c       -vdd*(szp*szp*rnu1+srp*srp*rnu2)
      b34=-(bp31*t14+bp32*t24+bp33*t34+bp34*t44)
      b41=-(bp41*t11+bp42*t21+bp43*t31+bp44*t41)
     c       +vdd*(upbar*upj*rnu1+vpbar*vpj*rnu2
     c       -(alpj-ei(i,j))*rnu3)
      b42=-(bp41*t12+bp42*t22+bp43*t32+bp44*t42)
     c      -vdd*( upbar*srp*rnu1+vpbar*szp*rnu2-uuj*rnu3)
      b43=-(bp41*t13+bp42*t23+bp43*t33+bp44*t43)
     c      -vdd*(-upbar*szp*rnu1+vpbar*srp*rnu2-vvj*rnu3)
      b44=-(bp41*t14+bp42*t24+bp43*t34+bp44*t44)
     c      -vdd*rnu3
      if (j.eq.2) then
      d1=aa*(1.0-vpj/ccj)*sj*dt/vol(i,j)
      bp1=(alpj*beta+vpj*ccj)*d1
      bp2=(-uuj*beta-szp*ccj)*d1
      bp3=(-vvj*beta-srp*ccj)*d1
      bp4=beta*d1
      b21=b21+szp*bp1
      b22=b22+szp*bp2
      b23=b23+szp*bp3
      b24=b24+szp*bp4
      b31=b31+srp*bp1
      b32=b32+srp*bp2
      b33=b33+srp*bp3
```

```
      b34=b34+srp*bp4
      endif
      a11=1.-b11-vr*(c11+a11)
      a12=  -b12-vr*(c12+a12)
      a13=  -b13-vr*(c13+a13)
      a14=  -b14-vr*(c14+a14)
      a21=  -b21-vr*(c21+a21)
      a22=1.-b22-vr*(c22+a22)
      a23=  -b23-vr*(c23+a23)
      a24=  -b24-vr*(c24+a24)
      a31=  -b31-vr*(c31+a31)
      a32=  -b32-vr*(c32+a32)
      a33=1.-b33-vr*(c33+a33)
      a34=  -b34-vr*(c34+a34)
      a41=  -b41-vr*(c41+a41)
      a42=  -b42-vr*(c42+a42)
      a43=  -b43-vr*(c43+a43)
      a44=1.-b44-vr*(c44+a44)
      iflag=2
      go to 6
    8 continue
      cp11= 1.0
      cp12= 0.0
      cp13= 0.5*rccsq
      cp14= 0.5*rccsq
      cp21= uu
      cp22= srp*rr
      cp23= (uu+szp*cc)*0.5*rccsq
      cp24= (uu-szp*cc)*0.5*rccsq
      cp31= vv
      cp32=-szp*rr
      cp33= (vv+srp*cc)*0.5*rccsq
      cp34= (vv-srp*cc)*0.5*rccsq
      cp41= alp
      cp42= rr*up
      cp43= 0.5*((alp+vp*cc)*rccsq+1.0/beta)
      cp44= 0.5*((alp-vp*cc)*rccsq+1.0/beta)
      vdd=vdd*(p(i,j)/(gm1*ei(i,j)))/rrj
      rnu1=ttc*(1.0+tt*ttu)*rmu
      rnu2=ttc*(1.0+tt)*(rlmbda+2.0*rmu)
      rnu3=ttc*(1.0+tt*tte)*rk
      c11=-(cp11*t11+cp12*t21+cp13*t31+cp14*t41)
      c12=-(cp11*t12+cp12*t22+cp13*t32+cp14*t42)
      c13=-(cp11*t13+cp12*t23+cp13*t33+cp14*t43)
      c14=-(cp11*t14+cp12*t24+cp13*t34+cp14*t44)
      c21=-(cp21*t11+cp22*t21+cp23*t31+cp24*t41)
     c      +vdd*( srp*upj*rnu1+szp*vpj*rnu2)
      c22=-(cp21*t12+cp22*t22+cp23*t32+cp24*t42)
     c      -vdd*(srp*srp*rnu1+szp*szp*rnu2)
      c23=-(cp21*t13+cp22*t23+cp23*t33+cp24*t43)
     c      +vdd*szp*srp*(rnu1-rnu2)
      c24=-(cp21*t14+cp22*t24+cp23*t34+cp24*t44)
```

```
      c31=-(cp31*t11+cp32*t21+cp33*t31+cp34*t41)
     c       +vdd*(-szp*upj*rnu1+srp*vpj*rnu2)
      c32=-(cp31*t12+cp32*t22+cp33*t32+cp34*t42)
     c       +vdd*szp*srp*(rnu1-rnu2)
      c33=-(cp31*t13+cp32*t23+cp33*t33+cp34*t43)
     c       -vdd*(szp*szp*rnu1+srp*srp*rnu2)
      c34=-(cp31*t14+cp32*t24+cp33*t34+cp34*t44)
      c41=-(cp41*t11+cp42*t21+cp43*t31+cp44*t41)
     c       +vdd*(upbar*upj*rnu1+vpbar*vpj*rnu2
             -(alpj-ei(i,j-1))*rnu3)
      c42=-(cp41*t12+cp42*t22+cp43*t32+cp44*t42)
     c       -vdd*( upbar*srp*rnu1+vpbar*szp*rnu2-uuj*rnu3)
      c43=-(cp41*t13+cp42*t23+cp43*t33+cp44*t43)
     c       -vdd*(-upbar*szp*rnu1+vpbar*srp*rnu2-vvj*rnu3)
      c44=-(cp41*t14+cp42*t24+cp43*t34+cp44*t44)
     c       -vdd*rnu3
      if (j.eq.jl) then
      dl=-aa*(1.0+vpj/ccj)*sj*dt/vol(i,j)
      cp1=(alpj*beta)*dl
      cp2=(-uuj*beta)*dl
      cp3=(-vvj*beta)*dl
      cp4=beta*dl
      c21=c21+szp*cp1
      c22=c22+szp*cp2
      c23=c23+szp*cp3
      c24=c24+szp*cp4
      c31=c31+srp*cp1
      c32=c32+srp*cp2
      c33=c33+srp*cp3
      c34=c34+srp*cp4
      go to 10
      endif
    9 continue
      if (j.eq.jl) go to 10
      call impli(dt,i,j,iadd)
      r11=a11
      r21=a21
      r31=a31
      r41=a41
      s12=a12/r11
      s13=a13/r11
      s14=a14/r11
      r22=a22-r21*s12
      r32=a32-r31*s12
      r42=a42-r41*s12
      s23=(a23-r21*s13)/r22
      s24=(a24-r21*s14)/r22
      r33=a33-r31*s13-r32*s23
      r43=a43-r41*s13-r42*s23
      s34=(a34-r31*s14-r32*s24)/r33
      r44=a44-r41*s14-r42*s24-r43*s34
      yy1=du(i,j,1)/r11
```

```
      yy2=(du(i,j,2)-r21*yy1)/r22
      yy3=(du(i,j,3)-r31*yy1-r32*yy2)/r33
      yy4=(du(i,j,4)-r41*yy1-r42*yy2-r43*yy3)/r44
      yy3=yy3-s34*yy4
      yy2=yy2-s23*yy3-s24*yy4
      yy1=yy1-s12*yy2-s13*yy3-s14*yy4
      du(i,j,1)=yy1
      du(i,j,2)=yy2
      du(i,j,3)=yy3
      du(i,j,4)=yy4
      if (j.eq.2) go to 10
      yy1=c11/r11
      yy2=(c21-r21*yy1)/r22
      yy3=(c31-r31*yy1-r32*yy2)/r33
      g41(j)=(c41-r41*yy1-r42*yy2-r43*yy3)/r44
      g31(j)=yy3-s34*g41(j)
      g21(j)=yy2-s23*g31(j)-s24*g41(j)
      g11(j)=yy1-s12*g21(j)-s13*g31(j)-s14*g41(j)
      yy1=c12/r11
      yy2=(c22-r21*yy1)/r22
      yy3=(c32-r31*yy1-r32*yy2)/r33
      g42(j)=(c42-r41*yy1-r42*yy2-r43*yy3)/r44
      g32(j)=yy3-s34*g42(j)
      g22(j)=yy2-s23*g32(j)-s24*g42(j)
      g12(j)=yy1-s12*g22(j)-s13*g32(j)-s14*g42(j)
      yy1=c13/r11
      yy2=(c23-r21*yy1)/r22
      yy3=(c33-r31*yy1-r32*yy2)/r33
      g43(j)=(c43-r41*yy1-r42*yy2-r43*yy3)/r44
      g33(j)=yy3-s34*g43(j)
      g23(j)=yy2-s23*g33(j)-s24*g43(j)
      g13(j)=yy1-s12*g23(j)-s13*g33(j)-s14*g43(j)
      yy1=c14/r11
      yy2=(c24-r21*yy1)/r22
      yy3=(c34-r31*yy1-r32*yy2)/r33
      g44(j)=(c44-r41*yy1-r42*yy2-r43*yy3)/r44
      g34(j)=yy3-s34*g44(j)
      g24(j)=yy2-s23*g34(j)-s24*g44(j)
      g14(j)=yy1-s12*g24(j)-s13*g34(j)-s14*g44(j)
   10 continue
      do 15 j=3,jl-1
      du1=du(i,j-1,1)
      du2=du(i,j-1,2)
      du3=du(i,j-1,3)
      du4=du(i,j-1,4)
      du(i,j,1)=du(i,j,1)-g11(j)*du1-g12(j)
     c          *du2-g13(j)*du3-g14(j)*du4
      du(i,j,2)=du(i,j,2)-g21(j)*du1-g22(j)
     c          *du2-g23(j)*du3-g24(j)*du4
      du(i,j,3)=du(i,j,3)-g31(j)*du1-g32(j)
     c          *du2-g33(j)*du3-g34(j)*du4
      du(i,j,4)=du(i,j,4)-g41(j)*du1-g42(j)
```

```
     c          *du2-g43(j)*du3-g44(j)*du4
   15 continue
   20 continue
      return
      end

c
c-----------------------------------------------------------------------
      subroutine prntff(time)
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      common/b/iturb,prt,ep(100,100)
      common/tem/t(101,101),ttwall
      common/speed/rmach(100,100)

      write(16,10)
   10 format(1h1,50x,27h*   f l o w   f i e l d   *,//)
      write(16,20) time,nadv
   20 format(34x,7htime = ,e14.7,2x,7hnadv = ,i9)

c
       write(50,20) time,nadv
c

      do 30 i=1,il,2
      ip1=i+1
c      write(16,24)i,ip1
c    24 format(1h0,8x,8hcolumn  ,i5,45x,8hcolumn  ,i5)

c      write(16,25)
c    25 format(1h ,3x,1hj,5x,3hrho,9x,1hv,10x,1hu
c     c      ,10x,1hp,9x,2hei,12x,
c     c      3hrho,9x,1hv,10x,1hu,10x,1hp,9x,3hei )
      do 29 j=1,jl
c      write(16,27) j,rho(i,j,m),v(i,j),u(i,j),p(i,j),ei(i,j),
c     c             rho(ip1,j,m),v(ip1,j),u(ip1,j)
c     c            ,p(ip1,j),ei(ip1,j)

c     -insert another write statement for graphics file-
c       write(50,27) j,rho(i,j,m),v(i,j),u(i,j),p(i,j),ei(i,j),
c     c             rho(ip1,j,m),v(ip1,j),u(ip1,j)
c     c             ,p(ip1,j),ei(ip1,j)
c
         write(50,27) j,rho(i,j,m),v(i,j),u(i,j),p(i,j),t(i,j),
```

```
      c       rmach(i,j),rho(ip1,j,m),v(ip1,j),u(ip1,j),p(ip1,j),
      c       t(ip1,j),rmach(ip1,j)
   27 format(1h ,i5,6e11.4,3x,6e11.4)
c
   29 continue
   30 continue
      return
      end
c
c-----------------------------------------------------------------------
      subroutine prntxy
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      ilp1=il+1
      jlp1=jl+1
c      write(16,10)
c   10 format(1h1,50x,38h*   m e s h   c o o r i n a t e s   *,//)
      do 30 i=1,ilp1,4
      ip1=i+1
      ip2=i+2
      ip3=i+3
c      write(16,20) i,ip1,ip2,ip3
c   20 format(1h0,8x,3(8hcolumn  ,i5,16x),8hcolumn  ,i5,/,
c     c       1h ,4x,1hj,6x,3(1hx,11x,1hy,16x),1hx,11x,1hy)
      do 30 j=1,jlp1
c      write(16,25) j,z(i,j),r(i,j),z(ip1,j),r(ip1,j),z(ip2,j),
c     c             r(ip2,j),z(ip3,j),r(ip3,j)
   25 format(1h ,i5,3(2e13.6,3x),2e13.6)

c     -insert another write statement for graphics files-

        write(50,25) j,z(i,j),r(i,j),z(ip1,j),r(ip1,j),z(ip2,j),
     c              r(ip2,j),z(ip3,j),r(ip3,j)
c
   30 continue
      return
      end
c
c-----------------------------------------------------------------------
      subroutine turbf1
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
```

```
      c          u(100,100),v(100,100),ei(100,100),p(100,100),
      c          fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
      c          gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
      c          il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
      c          iadbwl,eiwall,lcnt,cfl,
      c          nadv,nend,nvisc
       common/b/iturb,prt,ep(100,100)
       dimension yi(100),aomega(100),yyj(100)
       common/gm/thrt

       dd=2.*thrt
       aplus=26.0
       ccp=1.6
       ckleb=0.3
       cwk=0.25
       smlk=0.4
       bigk=0.0168
       prt=0.9
       aa=0.80
       bb=-7.10e-5
c      cmutm=14.0       to be used to simulate transition
       do 10 i=1,il
       sj=sqrt((z(i+1,2)-z(i,2))**2+(r(i+1,2)-r(i,2))**2)
       szp=-(r(i+1,2)-r(i,2))/sj
       srp= (z(i+1,2)-z(i,2))/sj
       u2max=u(i,2)*u(i,2)+v(i,2)*v(i,2)
       dudymx=2.0*abs(srp*u(i,2)-szp*v(i,2))*sj/vol(i,2)
       do 5 j=2,jlm1
       dy=0.5*abs(szp*(z(i,j+2)-z(i,j))+srp*(r(i,j+2)-r(i,j)))
       dudy=abs(srp*(u(i,j+1)-u(i,j))-szp*(v(i,j+1)-v(i,j)))/dy
       if (dudy.gt.dudymx) dudymx=dudy
       u2=u(i,j+1)*u(i,j+1)+v(i,j+1)*v(i,j+1)
       if (u2.gt.u2max) u2max=u2
     5 continue
       temp=0.5*(ei(i,1)+ei(i,2))/cv
c       rmu=1.458e-6*sqrt(temp**3)/(temp+110.4)
c      - uf4 property
c        rmu=3.357e-6*sqrt(temp)/(aa + bb*temp)
c     - UF4 viscosity at 2000 K
       rmu=8.67e-5
       ra=sqrt(rho(i,2,1)*rmu*dudymx)/(aplus*rmu)
c
         yyplus=5.0
         if(nadv.eq.nend)then
         yyj(i)=(1.e+3*yyplus)/(ra*aplus)
         endif

         ilm5=il-3
         if(nadv.eq.nend.and.i.eq.ilm5) then
         do 11 j=2,jlm1
         yj=abs(sxp*(z(ilm5,j+1)-z(iilm5,2))
      *     +syp*(r(ilm5,j+1)-r(ilm5,2)))
```

```
      yplus=ra*aplus*yj
c     - calculate emperical uplus using deissler's formula
      if (yplus.lt.5) then
      uuplus=yplus
      else if ( yplus.gt.5 . and. yplus.lt.26) then
      uuplus=-3.05 + 5.0*alog(yplus)
      else
      uuplus=(1./0.36)*alog(yplus)+3.8
c      uuplus=5.5 + 2.5*alog(yplus)
      endif

      uj=0.5*(u(ilm5,j+1)+u(ilm5,j))
      uplus=uj/sqrt(rmu*dudymx/rho(ilm5,2,1))
      write(22,*) yplus,uplus,uuplus
c     write(*,*) nadv,yj,yplus,uplus
11    continue
      endif
      fmax=0.0
      ymax=1.0
      iflag=0
    do 7 j=2,jlm1
    ii=i
    if (ii.eq.1) ii=2
    if (ii.eq.il) ii=il-1
    due=0.5*(u(ii+1,j)-u(ii-1,j))
    dve=0.5*(v(ii+1,j)-v(ii-1,j))
    dze=z(i+1,j+1)-z(i,j+1)
    dre=r(i+1,j+1)-r(i,j+1)
    dun=u(i,j+1)-u(i,j)
    dvn=v(i,j+1)-v(i,j)
    dzn=0.25*(z(i+1,j+2)+z(i,j+2)-z(i+1,j)-z(i,j))
    drn=0.25*(r(i+1,j+2)+r(i,j+2)-r(i+1,j)-r(i,j))
    dzr=1.0/(dze*drn-dzn*dre)
    dvz= (dve*drn-dvn*dre)*dzr
    dur=-(due*dzn-dun*dze)*dzr
    yj=abs(szp*(z(i,j+1)-z(i,2))+srp*(r(i,j+1)-r(i,2)))
    yi(j)=yj
    aomega(j)=abs(dur-dvz)
    fj=yj*aomega(j)*(1.0-exp(-yj*ra))
    if (fj.gt.fmax.and.j.le.jlfm) then
    fmax=fj
    ymax=yj
    iflag=1
    endif
  7 continue

    fwake=ymax*fmax
    if (iflag.eq.1) then
    fwakep=cwk*ymax*u2max/fmax
    if (fwakep.lt.fwake) fwake=fwakep
    endif
    jep=jl
```

```
      do 8 j=2,jlm1
      yj=yi(j)
      epi=rho(i,j,1)*((smlk*yj)*(1.0-exp(-yj*ra)))**2*aomega(j)
      fkleb=1.0/(1.0+5.5*(ckleb*yj/ymax)**6)
      epo=bigk*ccp*rho(i,j,1)*fwake*fkleb
      ep(i,j)=amin1(epi,epo)
      if (j.lt.jep.and.epi.gt.epo) jep=j
      if (j.gt.jep) ep(i,j)=epo
    8 continue
      ep(i,1)=0.0
      ep(i,jl)=ep(i,jl-1)
   10 continue
      if(nadv.eq.nend)then
        do 100 i=2,il-2
        zd=0.5*(z(i,2)+z(i+1,2))/dd
        write(999,*) zd,yyj(i)
100     continue
        endif
      return
      end
c
c-----------------------------------------------------------------------
      subroutine rdwrt(ise,time)
c-----------------------------------------------------------------------
c
c     subroutine to read in data from previous runs or write
c     data.
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      common/e/g11(101),g12(101),g13(101),g14(101),
     c         g21(101),g22(101),g23(101),g24(101),
     c         g31(101),g32(101),g33(101),g34(101),
     c         g41(101),g42(101),g43(101),g44(101),
     c         ff(101),implt
      if (ise.eq.2) go to 20
      read(3,*) nadv,time,il,jl,vou,ptt,ttt,pe
      read(3,*) ((rho(i,j,1),u(i,j),v(i,j),ei(i,j),p(i,j),
     c            vol(i,j),i=1,il),j=1,jl)
      ilp1=il+1
      jlp1=jl+1
      read(3,*) ((z(i,j),r(i,j),i=1,ilp1),j=1,jlp1)
      do 10 j=1,jl
      do 10 i=1,il
      rho(i,j,2)=rho(i,j,1)
```

```
      rhou(i,j,1)=rho(i,j,1)*u(i,j)
      rhov(i,j,1)=rho(i,j,1)*v(i,j)
      rhou(i,j,2)=rhou(i,j,1)
      rhov(i,j,2)=rhov(i,j,1)
      e(i,j,1)=rho(i,j,1)*(ei(i,j)+0.5*(u(i,j)*u(i,j)
     1                                  +v(i,j)*v(i,j)))
      e(i,j,2)=e(i,j,1)
   10 continue
      return
   20 continue
      write(4,*) nend,time,il,jl,vou,ptt,ttt,pe
      write(4,*) ((rho(i,j,1),u(i,j),v(i,j),ei(i,j),p(i,j),
     c              vol(i,j),i=1,il),j=1,jl)
      ilp1=il+1
      jlp1=jl+1
      write(4,*) ((z(i,j),r(i,j),i=1,ilp1),j=1,jlp1)
      return
      end

c-----------------------------------------------------------------------
      subroutine fsifs(i,j,ii)
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
     c         nadv,nend,nvisc
      common/splt/iflxs,sz,sr

      si=sqrt(sz*sz+sr*sr)
      szp=sz/si
      srp=sr/si
      rr=rho(ii,j,n)
      uu=u(ii,j)
      vv=v(ii,j)
      up= szp*uu+srp*vv
      vp=-srp*uu+szp*vv
      cc=sqrt(ggm1*abs(ei(ii,j)))
      alp=0.5*(up*up+vp*vp)
      do 10 nn=1,2
      i1=i+2-nn
      i2=i1+3-2*nn
      if (iflxs.eq.1.or.i.le.nn.or.i.ge.il+nn-3) i2=i1
      if (abs(p(i2,j)-p(i1,j)).gt.
     c    0.5*amin1(abs(p(i1,j)),abs(p(i1,j)))) i2=i1
      if (abs(p(i+1,j)-p(i,j)).gt.
     c    0.5*amin1(abs(p(i+1,j)),abs(p(i,j)))) then
      i2=i1
```

```
      rr=rho(i1,j,n)
      uu=u(i1,j)
      vv=v(i1,j)
      up= szp*uu+srp*vv
      vp=-srp*uu+szp*vv
      cc=sqrt(ggm1*abs(ei(i1,j)))
      alp=0.5*(up*up+vp*vp)
      endif
      dr=1.5*rho(i1,j,n)-0.5*rho(i2,j,n)
      dru=1.5*rhou(i1,j,n)-0.5*rhou(i2,j,n)
      drv=1.5*rhov(i1,j,n)-0.5*rhov(i2,j,n)
      de=1.5*e(i1,j,n)-0.5*e(i2,j,n)
      dp=(dr*alp-uu*dru-vv*drv+de)*gm1

      xx1=dr-dp/(cc*cc)
      xx2=(-up*dr+szp*dru+srp*drv)*cc+dp
      xx3=(-vp*dr-srp*dru+szp*drv)/rr
      xx4=-xx2+2.0*dp

      dd1=si*up
      dd2=si*(up+cc)
      dd4=si*(up-cc)

      sgns=2*nn-3

      yy1=0.5*(dd1+sgns*abs(dd1))*xx1
      yy2=0.5*(dd2+sgns*abs(dd2))*xx2
      yy3=0.5*(dd1+sgns*abs(dd1))*xx3
      yy4=0.5*(dd4+sgns*abs(dd4))*xx4

      t0=yy1+0.5*(yy2+yy4)/(cc*cc)
      t1=up*t0+0.5*(yy2-yy4)/cc
      t2=vp*t0+rr*yy3
      fs1(k2)=(nn-1)*fs1(k2)+t0
      fs2(k2)=(nn-1)*fs2(k2)+szp*t1-srp*t2
      fs3(k2)=(nn-1)*fs3(k2)+srp*t1+szp*t2
      fs4(k2)=(nn-1)*fs4(k2)+
     c       alp*t0+rr*vp*yy3+0.5*(up*(yy2-yy4)/cc+(yy2+yy4)/gm1)
   10 continue
      return
      end
c-----------------------------------------------------------------------
      subroutine fsjfs(i,j,jj)
c-----------------------------------------------------------------------
c
      common/a/rho(100,100,2),rhou(100,100,2),rhov(100,100,2),
     c         e(100,100,2),fs1(2),fs2(2),
     c         u(100,100),v(100,100),ei(100,100),p(100,100),
     c         fs3(2),fs4(2),z(101,101),r(101,101),vol(101,101),
     c         gamma,gm1,ggm1,cv,pr,vou,ptt,ttt,pe,
     c         il,jl,ilm1,jlm1,jlfm,n,m,k1,k2,
     c         iadbwl,eiwall,lcnt,cfl,
```

```
c          nadv,nend,nvisc
 common/splt/iflxs,sz,sr
 if (j.eq.1.or.j.eq.jlm1) return
 sj=sqrt(sz*sz+sr*sr)
 szp=sz/sj
 srp=sr/sj

 rr=rho(i,jj,n)
 uu=u(i,jj)
 vv=v(i,jj)
 cc=sqrt(ggm1*abs(ei(i,jj)))

 do 10 nn=1,2

 up= srp*uu-szp*vv
 vp= szp*uu+srp*vv
 alp=0.5*(up*up+vp*vp)

 j1=j+2-nn
 j2=j1+3-2*nn
 if (iflxs.eq.1.or.j.le.nn.or.j.ge.jl+nn-3) j2=j1
 if (abs(p(i,j2)-p(i,j1)).gt.
c    0.5*amin1(abs(p(i,j1)),abs(p(i,j2)))) j2=j1
 if (abs(p(i,j+1)-p(i,j)).gt.
c    0.5*amin1(abs(p(i,j+1)),abs(p(i,j)))) then
 j2=j1
 rr=rho(i,j1,n)
 uu=u(i,j1)
 vv=v(i,j1)
 up= srp*uu-szp*vv
 vp= szp*uu+srp*vv
 cc=sqrt(ggm1*abs(ei(i,j1)))
 alp=0.5*(up*up+vp*vp)
 endif

 dr=1.5*rho(i,j1,n)-0.5*rho(i,j2,n)
 dru=1.5*rhou(i,j1,n)-0.5*rhou(i,j2,n)
 drv=1.5*rhov(i,j1,n)-0.5*rhov(i,j2,n)
 de=1.5*e(i,j1,n)-0.5*e(i,j2,n)
 dp=(alp*dr-uu*dru-vv*drv+de)*gm1

 wt1=1.0
 wt2=0.0
 if (j.lt.jlfm) then
 wt1=((z(i,j)-z(i,2))**2+(r(i,j)-r(i,2))**2)/
c    ((z(i,jlfm)-z(i,2))**2+(r(i,jlfm)-r(i,2))**2)
 wt2=1.0-wt1
 dr=wt1*dr+wt2*rho(i,j1,n)
 dru=wt1*dru+wt2*rhou(i,j1,n)
 drv=wt1*drv+wt2*rhov(i,j1,n)
 de=wt1*de+wt2*e(i,j1,n)
 dp=wt1*dp+wt2*p(i,j1)
```

```
      endif

      xx1=dr-dp/(cc*cc)
      xx2=(-up*dr+srp*dru-szp*drv)/rr
      xx3=(-vp*dr+szp*dru+syr*drv)*cc+dp
      xx4=-xx3+2.0*dp

      dd1=sj*vp
      dd3=sj*(vp+cc)
      dd4=sj*(vp-cc)

      sgns=2*nn-3

      yy1=0.5*(dd1+sgns*abs(dd1))*xx1
      yy2=0.5*(dd1+sgns*abs(dd1))*xx2
      yy3=0.5*(dd3+sgns*abs(dd3))*xx3
      yy4=0.5*(dd4+sgns*abs(dd4))*xx4

      vpj=szp*u(i,j1)+srp*v(i,j1)
      vp=wt1*vp+wt2*vpj
      alp=0.5*(up*up+vp*vp)

      t0=yy1+0.5*(yy3+yy4)/(cc*cc)
      t1=up*t0+rr*yy2
      t2=vp*t0+0.5*(yy3-yy4)/cc
      fs1(k2)=(nn-1)*fs1(k2)+t0
      fs2(k2)=(nn-1)*fs2(k2)+srp*t1+szp*t2
      fs3(k2)=(nn-1)*fs3(k2)-szp*t1+srp*t2
      fs4(k2)=(nn-1)*fs4(k2)+
     c  alp*t0+rr*up*yy2+0.5*(vp*(yy3-yy4)/cc+(yy3+yy4)/gm1)
   10 continue
      return
      end
```

REFERENCES

1. Diaz, N. J., Anghaie, S., Dugan, E. T., and Maya, I., Ultrahigh Temperature Reactor and Energy Conversion Research Program, INSPI Proposal to AFWAL, 1988.

2. Incropera, F. P and Dewitt, D. P., Introduction to Heat Transfer, John Wiley & Sons Inc., 1985, pp. 359-361.

3. Colburn, A. P., "A Method of Correlating Forced Convection Heat Transfer Data and a Comparison with Fluid Friction," Trans. A. I. Ch. E., Vol. 29, 174, 1933.

4. Sieder, E. N., and Tate, C. E., "Heat Transfer and Pressure Drop of Liquids in Tubes," Ind. Eng. Chem., Vol. 28, 1429, 1936.

5. Zellnik, H. E., and Churchill, S. W., "Convective Heat Transfer High Temperature Air Inside a Tube," A. I. Ch. E. Journal, Vol. 4, No. 1, 1958.

6. Viegas, J. R., Rubesin, M. W., and Hortsmann, C. C., "On the Use of Wall Function as Boundary Conditions for Two-Dimensional, Separated Compressible Flows," AIAA Paper No. 85-0180, 1985.

7. Ciofalo, J. R., and Collins, M. W., "k-ε Predictions of Heat Transfer in Turbulent Recirculating Flows Using an Improved Wall Treatment," Numerical Heat Transfer, Part B, Vol. 15, 1989, pp. 21-47.

8. Dijilali, N., Gartshore, I., and Salcudean, M., "Calculation of Convective Heat Transfer in Recirculating Turbulent Flow Using Various Near-Wall Turbulence Models," Numerical Heat Transfer, Part A, Vol. 16, 1989, pp. 189-212.

9. Launder, B. E., "On the Computation of Convective Heat Transfer in Complex Turbulent Flows," ASME J. of Heat Transfer, Vol. 110, 1988, pp. 1112-1128.

10. Launder, B. E., and Spalding, D. B., "The Numerical Computation of Turbulent Flows," Computer Methods in Applied Mechanics and Engineering, Vol. 3, 1974, pp. 269-289.

11. Rose, W. C., "Numerical Investigation in Three-Dimensional Internal Flows," Semiannual Status Report, NASA-CR-183108, March, 1988.

12. Jennings, M. J., and Mord, T., "Multidimensional Modeling of Convective Heat Transfer with Application to I. C. Engine," DOE/AL/33183 - T11, 1989.

13. Briley, W. R., and McDonald, H., "Solution of the Three-Dimensional Compressible Navier-Stokes Equations by an Implicit Technique," Proceeding of the 4th International Conference on Numerical Methods in Fluid Dynamics, No. 35, New York, 1975.

14. Warming, R. F., and Beam, R. M., "On the Construction of Implicit Factored Schemes for Conservation Laws," SIAM - AMS Proceedings, Vol. 11, 1978, pp. 85-129.

15. MacCormack, R. W., "A Numerical Method for Solving the Equations of Compressible Viscous Flow," AIAA Journal, Vol. 20, No. 9, 1982, pp. 1275-1281.

16. MacCormack, R. W., "Current Status of Numerical Solutions of the Navier-Stokes Equations," AIAA Paper, AIAA-85-0032, Proceeding of AIAA 23rd Aerospace Science Meeting, January 14-17, Reno, Nevada, 1985.

17. Baldwin, B. S., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper, AIAA-78-257, Proceeding of AIAA 16th Aerospace Science Meeting, January 16-18, Huntsville, Alabama, 1978.

18. Diamant, E. S., and Gore, L. A., Hammitt, A. G., and Hunter, H. M., "Feasibility Study of Direct Flow Gaseous Core Reactor Syatem," Contract Report, N66-16525, January, Redondo Beach, California, 1966.

19. Dam, H. V., and Hoogenboom, J. E., "Physics of a Gaseous Core Reactor," Nuclear Technology, Vol. 63, 1983, pp. 359-368.

20. Kerkdijk, C. B. W., and Kistmaker, J., "Temperature Profile and Power Calculation in Gas Core Reactor," N78-26837, 1979, pp. 1958-1961.

21. Hoogenboom, J. E., Van Dam, H., Kuijper, J. C., Kistemaker, J., Boersma, W., and Vitalis, F., "The Temperature Distribution in a Gas Core Fission Reactor," Ann. Nucl. Energy, Vol. 18, No. 4, 1991, pp. 183-195.

22. Watanabe, Y., Appelbaum, I., Diaz, N., and Maya, I., "Heat Transfer Analysis of Fuel Assemblies in A Heterogeneous Gas Core Nuclear Rocket," AIAA Paper No. 91-3517, 1991.

23. Kazanskii, K. A. and Novikov, V. M., "Thermophysical and Electrophysical Properties of Uranium Hexafluoride at Temperature of (1-11) x 10^3 K and Pressure of 0.1-100 atm," Translated from Teplofizika Vysokikh Temperatur, Vol. 14, No. 3, 1976, pp. 450-456.

24. Bird, R. B., Stewart, W. E., and Lightfoot, E. N., Transport Phenomena, John Wiely & Sons, Inc., New York, 1960.

25. Anderson, D. A., Tannehille, J. C., and Pletch, R. H., Computational Fluid Mechanics and Heat Transfer, Hemisphere Publishing Co., New York, 1984.

26. Bacher, W. and Karlsruhe, K., Gmelin Handbuch der Anorganischen Chemie, Spring-Verlag Inc., Berlin, 1980.

27. Oliver, C. C. and Dugan, E. T., "Thermophysical Properties of UF_6-He Mixture Relevant to Circulating Gas Core Reactor Systems," Nuclear Technology, Vol. 69, 1985.

28. Chapman, S. and Cowling, T. G., The Mathematical Theory of Non Uniform Gases, Cambridge University Press, London, 1939.

29. Siegel, R. and Howell, J. R., Thermal Radiation Heat Transfer, Hemisphere Publishing Co., New York, 1981.

30. Van Dem, H. and Hoogenboom, J. E., "Physics of A Gaseous Core Reactor," Nuclear Technology, Vol. 63, 1983, pp. 359-368.

31. Depoorter, G. L. and Rofer-Depooter, C. K., "The Absorption Spectrum of UF_6 from 2000 to 4200 Å," Spectroscopy Letter, 8(8), 1975, pp. 521-524.

32. Path, R. "Status of Opacity for Application to Uranium-Fueled Gas-Core Reactor," in Symp. on Research on Uranium Plasma and Their Technology Applications, Jan. 7-9, NASA SP 236, 1970, pp. 165-171.

33. Viskanta, R. and Grosh, R. J., "Boundary Layer in Thermal Radiation Absorbing and Emitting Media," Int. J. Heat Mass Transfer, Vol. 5, 1962, pp. 759-806.

34. Knight, D. D., "A Hybrid Explicit-Implicit Numerical Algorithm for the Three-Dimensional Compressible Navier-Stokes Equations," AIAA Journal, Vol. 22, No. 8, 1984, pp. 1056-1063.

35. MacCormack, R. W., "The Effect of Viscosity in Hypervelocity Impact Cratering," AIAA Paper No. 69-354, 1969.

36. Steger, J. and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamics Equations with Application to Finite Difference Methods," NASA TM-78605, 1982.

37. Chakravarthy, S. R., "Relaxation Methods for Unfactored Implicit Upwind Scheme," AIAA Paper No. 84-0165, 1984.

38. Napolitano, M. and Walters, R. W., "An Incremental Block-Line-Gauss-Seidel Method for the Navier-Stokes Equations," AIAA Paper No. 85-0033, 1985.

39. Zha, G., Liu, D., and Ma, T., "An Efficient Upwind/Relaxation Algorithm for the Euler and Navier-Stokes Equations," Int. Journal for Numerical Methods in Fluids, Vol. 9, 1989, pp. 517-529.

40. Schlichting, H., Boundary Layer Theory, McGraw-Hill Book Co., New York, 1979.

BIOGRAPHICAL SKETCH

Se-Won Chung was born on January 11, 1959, in Kyoungnam, Republic of Korea. After graduation from Busan Dong High School in Busan, he entered the Hanyang University, Seoul, Korea where he received his Bachelor of Science degree in nuclear engineering (1982) and Master of Science degree (1984). In 1986, he entered the University of Wisconsin - Madison where he received another Master of Science degree in nuclear engineering and engineering physics in May, 1988. He has been working on the degree of Doctor of Philosophy since 1988, while being a research assistant at the University of Florida.

On April 21, 1984, he married Hyun-Mi Kim and now they have a lovely daughter, Juyun, and a wonderful son, Jinook.

He is a student member of the American Institute of Aeronautics and Astronautics and American Nuclear Society.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

S. Anghaie, Chairman
Professor of Nuclear
Engineering Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

N.J. Diaz
Professor of Nuclear
Engineering Sciences

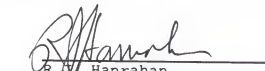
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

E.T. Dugan
Associate Professor of Nuclear
Engineering Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

R.B. Gaither for W.E. Lear
W.E. Lear
Assistant Professor of
Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



R. G. Hanrahan
Professor of Chemistry

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August, 1992

for Dean, College of Engineering

Dean, Graduate School